МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Закарпатський державний університет

Факультет інформатики

Кафедра програмного забезпечення автоматизованих систем

та фіз.-мат. дисциплін

Г.Е. Копча-Горячкіна

ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ В КОМБІНАТОРИЦІ ТА ГРАФАХ

Навчально-методичний посібник

частина І

Ужгород-2008

Копча-Горячкіна Галина Ернестівна

ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ В КОМБІНАТОРИЦІ ТА ГРАФАХ. Частина І: Навчально-методичний посібник для студентів факультету інформатики напряму „Комп'ютерні науки" зі спеціальності „Програмне забезпечення автоматизованих систем". – Ужгород: Видавництво Закарпатського державного університету, 2008 р.

Навчально-методичний посібник містить деякі теоретичні відомості, опис предмета навчальної дисципліни у відповідності до болонського процесу, навчально-тематичний план дисципліни «Фундаментальні алгоритми в комбінаториці та графах», зміст лекційних тем курсу, тем лабораторних занять та тем самостійної роботи студентів, перелік питань для модульних контролів, на залік, літературу.

Друкується за рішенням кафедри програмного забезпечення автоматизованих систем та фіз.-мат. дисциплін від 20.10.2008р., протокол № 2.

Вступ

В посібнику розглядаються деякі поняття, пов’язані з обчисленнями на дискретних скінчених математичних структурах, які часто називають комбінаторними обчисленнями. Ці обчислення потребують комбінаторного аналізу для встановлення властивостей і виявлення оцінки застосовності алгоритмів, що використовуються.

Дано опис предмета навчальної дисципліни у відповідності до болонського процесу, навчально-тематичний план дисципліни «Фундаментальні алгоритми в комбінаториці та графах», зміст лекційних тем курсу, тем лабораторних занять та тем самостійної роботи студентів, а також перелік питань на модуль І та модуль ІІ і на залік. З теоретичних питань включені поняття першої частини курсу, тобто фундаментальні алгоритми в комбінаториці. Алгоритми другої частини (на графах) будуть розглянуті в частині ІІ посібника з даної дисципліни.

Мета і завдання курсу

Ознайомити студентів з основними алгоритмами комбінаторного аналізу та теорії графів. Виробити вміння реалізовувати дані алгоритми на ЕОМ.

Опис предмета навчальної дисципліни

Курс: підготовка (бакалаврів, спеціалістів, магістрів)	**Напрям, спеціальності, освітньо-кваліфікаційний рівень**	**Характеристика навчальної дисципліни**
Кількість кредитів ECTS – 4,5 **Модулів**: к-ть модулів – 2 **Змістових модулів**: к-сть модулів – 2 **Загальна кількість годин** – 162 **Тижневих годин:** кількість годин – 2	**6.050103** **Програмна інженерія** **6.080400** **ПЗАС** **Спеціаліст**	**Дисципліна вільного вибору студента** **Рік підготовки** – 2008/2009 **Семестр** – 2 **Лекції** – 22 год. **Лабораторних** – 32 год. **Самостійна робота** – 108 год. **Вид підсумкового контролю** – залік

Навчально-тематичний план дисципліни «Фундаментальні алгоритми в комбінаториці та графах»

№ п/п	Назва теми, її зміст	Всього годин	Лекційні	Практичні заняття	Лабораторні роботи	Самостійна робота	Література
Змістовий модуль 1							
1.	Вступ. Основні поняття комбінаторики. Зв'язок з іншими дисциплінами	7	1	-	-	6	[1], с.48-50, [2], с.134-135
2.	Розміщення, сполучення, вибірки. Алгоритми їх обчислення	12	2	-	4	6	[1], с.50-51, [2], с.135-138
3.	Перестановки. Підстановки. Задача розкладання в ящики.	14	2	-	2	10	[1], с.51-53
4.	Алгоритм сортування „бульбашковим методом"	10	2	-	4	4	[2], с.141-142
5.	Алгоритм „Генерація перестановок"	20	2	-	8	10	[2], с.142-143
6.	Біном Ньютона. Властивості біноміальних коефіцієнтів	8	2	-	-	6	[1], с.53-55
7.	Генерація підмножин	5	1	-	4	-	[2], с.147-148
Модульний контроль (тестування, контрольна робота)							
Всього годин за модулем 1		**76**	**12**	**-**	**22**	**42**	
Змістовий модуль 2							
8.	Основні поняття теорії графів. Зв'язок з іншими дисциплінами	22	2	-	-	20	[2], с.189-197
9.	Представлення графів в ЕОМ. Способи задання графа	20	2	-	2	16	[2], с.201-203
10.	Обходи графів. Алгоритм пошуку в глибину в графі	16	2	-	4	10	[2], с.203-204
11.	Алгоритм пошуку в ширину в графі	16	2	-	4	10	[2], с.204-205
12.	Алгоритми пошуку найкоротших шляхів у графі	12	2	-	-	10	[2], с.228-233
Модульний контроль (тестування, контрольна робота)							
Всього годин за модулем 2		**86**	**10**	**-**	**10**	**66**	

Зміст лекційних тем курсу

1. Вступ. Основні поняття комбінаторики. Зв’язок з іншими дисциплінами. [1], с.48-50, [2], с.134-135.
2. Розміщення, сполучення, вибірки. Алгоритми їх обчислення. [1], с.50-51, [2], с.135-138.
3. Перестановки. Підстановки. Задача розкладання в ящики. [1], с.51-53.
4. Алгоритм сортування „бульбашковим методом”. [2], с.141-142.
5. Алгоритм „Генерація перестановок”. [2], с.142-143.
6. Біном Ньютона. Властивості біноміальних коефіцієнтів. [1], с.53-55.
7. Генерація підмножин. [2], с.147-148.
8. Основні поняття теорії графів. Зв’язок з іншими дисциплінами. [2], с.189-197.
9. Представлення графів в ЕОМ. Способи задання графа. [2], с.201-203.
10. Обходи графів. Алгоритм пошуку в глибину в графі. [2], с.203-204.
11. Алгоритм пошуку в ширину в графі. [2], с.204-205.
12. Алгоритми пошуку найкоротших шляхів у графі. [2], с.228-233.

Теми лабораторних занять

№ п/п	Назва теми	К-сть годин	Література
1.	Реалізація алгоритмів на обчислення розміщення та сполучення	4	[1], с.50-51, [2], с.135-138
2.	Реалізація алгоритму сортування „бульбашковим методом”	2	[2], с.141-142
3.	Реалізація алгоритму сортування елементів методом транспозиції (перестановкою сусідніх елементів)	2	[2], с.141-142
4.	Генерування перестановок в лексикографічному порядку	4	[2], с.142-143
5.	Генерування перестановок в антилексикографічному порядку	4	[2], с.142-143
6.	Генерування підмножин	2	[2], с.147-148
7.	Генерування сполучень	2	[2], с.147-148
8.	Визначення парності підстановки	2	[2], с.141
9.	**Змістовий модуль 1**		**Модульний контроль 1**
10.	Задання графа матрицею інцидентності	2	[2], с.201-203
11.	Реалізація алгоритму пошуку в глибину в графі	4	[2], с.203-204
12.	Реалізація алгоритму пошуку в ширину в графі	4	[2], с.204-205
13.	**Змістовий модуль 2**		**Модульний контроль 2**

Теми самостійної роботи студентів

№ п/п	Назва теми	К-сть годин
Змістовий модуль1		
1.	Задача про цілочислові розв’язки. Числа Стірлінга та числа Белла	6
2.	Властивості біноміальних коефіцієнтів	6
3.	Перестановки. Підстановки. Множення підстановок	6
4.	Обернена підстановка. Порядок підстановки	6
5.	Розклад підстановок в добуток циклів	6
6.	Розклад підстановок в добуток транспозицій	4
7.	Транспозиції, інверсії, парність підстановки	8
Література: [1], с.56-57, с.53-55, [2], с.136-143		
Всього за змістовий модуль 1		**42**
Змістовий модуль 2		
8.	Алгоритм Флойда	8
9.	Алгоритм Дейкстри	8
10.	Дерева. Основні властивості дерев.	10
11.	Орієнтовані, впорядковані і бінарні дерева	10
12.	Зображення дерев в ЕОМ.	10
13.	Обходи бінарних дерев. Алгоритм симетричного обходу бінарного дерева	10
14.	Дерева сортування. Способи реалізації асоціативної пам’яті	10
Література: [2], с.229-230, с.230-231, с.234-238, с.238-242, с.243-244, с.244-246, с.246-253		
Всього за змістовий модуль 2		**66**

Література

1. Нікольський Ю. В., Пасічник В.В., Щербина Ю.М. Дискретна математика, Київ, Видавнича група BHV, 2007.
2. Новиков Ф.А.Дискретная математика для программистов, учебник, Санкт-Петербург, Изд. дом «Питер», 2001.

Перелік питань на модуль І

1. Комбінаторика, основні правила комбінаторики, основні комбінаторні конфігурації.
2. Основні комбінаторні об'єкти. Вибірки, типи вибірок.
3. Розміщення, сполучення, типи сполучень.
4. Число розміщень без повторень, число сполучень без повторень.
5. Перестановки, підстановки.
6. Множення підстановок, властивості операції множення.
7. Порядок підстановки. Розклад підстановки в добуток циклів.
8. Обернена підстановка, обернений цикл.
9. Транспозиції, інверсії, парність підстановки.
10. Алгоритми сортування. Алгоритм сортування методом транспозиції (перестановкою сусідніх елементів).
11. Алгоритм сортування „бульбашковим методом”.

Перелік питань на модуль ІІ

1. Граф, порядок графа, суміжність, інцидентність.
2. Степінь вершини, відстань між вершинами.
3. Маршрут, ланцюг, простий ланцюг, геодезичний ланцюг.
4. Цикл, простий цикл, ярус.
5. Зв’язний граф, тривіальний граф, дерево.
6. Орієнтований маршрут, орієнтований ланцюг.
7. Задання графа за допомогою списку ребер та матриці інцидентності.
8. Задання орграфа за допомогою списку ребер та матриці інцидентності.
9. Ізоморфізм графів.
10. Основні алгоритми на графах: алгоритм пошуку в глибину, алгоритм пошуку в ширину.

Перелік питань на залік

1. Комбінаторика, основні правила комбінаторики, основні комбінаторні конфігурації.
2. Основні комбінаторні об'єкти. Вибірки, типи вибірок.
3. Розміщення, сполучення, типи сполучень.
4. Число розміщень без повторень, число сполучень без повторень.
5. Перестановки, підстановки.
6. Множення підстановок, властивості операції множення.
7. Порядок підстановки. Розклад підстановки в добуток циклів.

8. Обернена підстановка, обернений цикл.
9. Транспозиції, інверсії, парність підстановки.
10. Алгоритми сортування. Алгоритм сортування методом транспозиції (перестановкою сусідніх елементів).
11. Алгоритм сортування „бульбашковим методом”.
12. Генерування перестановок. Антилексикографічний порядок.
13. Алгоритм генерування перестановок в лексикографічному порядку.
14. Генерування підмножин. Алгоритм генерування n-елементних підмножин m-елементної множини.
15. Генерування сполучень. Алгоритм побудови лексикографічно наступного сполучення.
16. Граф, порядок графа, суміжність, інцидентність.
17. Степінь вершини, відстань між вершинами.
18. Маршрут, ланцюг, простий ланцюг, геодезичний ланцюг.
19. Цикл, простий цикл, ярус.
20. Зв’язний граф, тривіальний граф, дерево.
21. Орієнтований маршрут, орієнтований ланцюг.
22. Задання графа за допомогою списку ребер та матриці інцидентності.
23. Задання орграфа за допомогою списку ребер та матриці інцидентності.
24. Ізоморфізм графів.
25. Основні алгоритми на графах.
26. Алгоритм пошуку в глибину в графі.
27. Алгоритм пошуку в ширину в графі.

ОСНОВНІ АЛГОРИТМИ В КОМБІНАТОРИЦІ

На практиці люди часто зустрічаються із задачами, в яких необхідно підрахувати число всіх можливих способів розміщення деяких предметів скінченої множини або число всіх можливих способів виконання певної дії зі скінченої множини таких дій.

Наприклад.

1. Скількома способами можуть розподілитися глядачі на чемпіонаті світу з футболу серед 24-ти команд-учасниць фінальної групи?

2. З пункту *A* міста Києва в пункт *B* можна доїхати трьома видами транспорту:

- тролейбусом (Т),
- автобусом (А),
- метро (М);

з пункту *B* в пункт *C* – лише двома видами транспорту: тролейбусом (Т) і автобусом (А). Скількома способами можна доїхати з пункту *A* в пункт *C* міста Києва?

3. Агентство нерухомості має базу даних з *n* записів, причому кожен запис містить одну пропозицію (що є в наявності) і один запит (що потрібно). Потрібно знайти всі такі пари записів, в яких пропозиція першого запису співпадає з запитом другого запису і навпаки, пропозиція другої співпадає з запитом першої (тобто це називається підбором варіантів обліку).

4. Скількома різними способами можна розставити дужки у виразі *a+b+c+d+e+f*, якщо операція + асоціативна, а букви *a,b,c,d,e,f* – деякі числа?

На перший погляд здається, що формули для розв’язання цих задач корисні лише для розв'язання олімпіадних задач і не мають відношення до практичного програмування.

Насправді, це зовсім не так. Обчислення на дискретних скінчених математичних структурах, які часто називають комбінаторними обчисленнями, потребують комбінаторного аналізу для встановлення властивостей і виявлення оцінки застосовності алгоритмів, що використовуються.

Для прикладу розглянемо задачу **3**.

Нехай СУБД даного агентства нерухомості дозволяє перевірити варіант за одну мілісекунду. Тоді при “лобовому” алгоритмі пошуку варіантів (кожен запис порівнюється з кожним) потрібно $n(n-1)/2$ порівнянь.

Якщо $n=100$, то відповідь буде одержана за 4,95 секунди.

Але якщо $n=100\ 000$ (що досить реально), то відповідь буде одержано за 4 999 950 секунд – а це 1389 годин!

І це ми оцінили лише трудомісткість підбору прямих варіантів, а ще існують варіанти, коли кількість учасників угоди більше двох …

Отже, як бачимо, комбінаторні обчислення потребують попереднього аналізу і кількісної оцінки вихідних задач і алгоритмів, що використовуються.

Зазвичай задачі оцінюються з точки зору розміру, тобто загальної кількості різних варіантів, серед яких треба знайти розв'язок, а алгоритми оцінюються з точки зору складності.

При цьому розрізняють складність за часом (часову складність), тобто кількість необхідних кроків алгоритму, і складність за пам'яттю (або об'ємну складність), тобто об'єм пам'яті, необхідний для роботи алгоритму.

На практиці виникає необхідність підрахувати кількість можливих комбінацій об'єктів, що задовольняють певним умовам. Задачі, в яких визначають всі можливі різні комбінації, складені зі скінченого числа елементів за деяким правилом, називають комбінаторними (наші задачі **1-4**), а розділ математики, в якому вивчається їх розв'язання, – комбінаторикою, методи їх розв'язку – методами комбінаторного аналізу.

Таким чином, у комбінаториці (комбінаторному аналізі) вивчають об'єкти зі скінченої множини $A = \{a_1, a_2, a_n\}$ та їх властивості, а також визначають кількість об'єктів із певними властивостями.

Розглядають також принципи, що використовуються в різних задачах, на яких ґрунтуються важливі методи математичного доведення, широко застосовні в теорії скінчених автоматів та інших розділах.

Для формуліровки і розв'язку комбінаторних задач використовують різні моделі комбінаторних конфігурацій.

Найбільш популярні такі:

1.Дано n предметів. Їх треба розмістити по m ящиках так, щоб виконувались задані обмеження. Скількома способами це можна зробити?

2.Розглянемо множину функцій $F: X \to Y$, де $|X| = n$, $|Y| = m$, $X = \{1,..,n\}$.

Не обмежуючи загальності, можна вважати, що

$Y = \{1,..,m\}$, $F =< F(1),..., F(n) >$, $1 \le F(i) \le m$.

Скільки існує функцій *F*, що задовольняють заданим обмеженням?

Є два основні правила комбінаторики.

1.Правило суми. Якщо об'єкт *x* можна вибрати n_1 способами, а інший об'єкт $y - n_2$ способами, то можна вибрати або *x*, або y $n_1 + n_2$ способами.

Наприклад.

Студент має вибрати тему курсової роботи зі списку, розміщеного на трьох аркушах. Аркуші містять відповідно 20,15 і 17 тем. З якої кількості можливих тем студент робить свій вибір?

За правилом суми ця кількість: 20+15+17=52.

2.Правило добутку. Якщо об'єкт *x* можна вибрати n_1 способами та після кожного такого вибору об'єкт y можна вибрати n_2 способами, то пару об'єктів *(x,y)* у зазначеному порядку можна вибрати $n_1 n_2$ способами.

Або інакше: нехай деяку процедуру можна виконати, розв'язавши два завдання. Якщо є n_1 спосіб розв'язати перше завдання та n_2 способи розв'язати після цього друге завдання, то всю процедуру можна виконати $n_1 n_2$ способами.

Наприклад.

1. Розглянемо задачу **1**. Золоту медаль може одержати будь-яка з 24-х команд, тобто є 24 можливості. Срібну медаль може виграти одна з 23-х команд, бронзову – одна з 22-х команд.

Отже, за правилом добутку загальне число способів розподілу медалей 24·23·22=12144.

2. Розглянемо задачу **2**. Розв'язок цієї задачі зводиться до підрахунку числа елементів в декартовому добутку множин $\{A,T,M\}\times\{A,T\}$.

Кількість таких елементів дорівнює добутку числа елементів першої множини на число елементів другої множини, тобто 3·2=6.

Отже, існує шість способів доїхати з пункту *A* в пункт *C*.

3. Наступна задача.

Скільки тризначних чисел можна скласти з цифр 0,1,2,3,4,5, якщо:

1) цифри можуть повторюватись;
2) жодна з цифр не повторюється двічі;
3) цифри непарні і можуть повторюватися.

Розв'язок.

1) Першою цифрою може бути одна з цифр 1,2,3,4,5, оскільки 0 не може бути першою цифрою, бо в цьому випадку число не буде тризначним.

Якщо перша цифра вибрана, то друга може бути вибрана шістьма способами, як і третя.

Отже, загальне число тризначних цифр 5·6·6=180.

2) Першою цифрою може бути одна з цифр – 1,2,3,4,5. Якщо перша цифра вибрана, то другою може бути теж одна з п'яти цифр (тут уже враховується 0), а третя може бути вибрана чотирма способами з 4-х цифр, що залишились. Отже, загальна кількість 5·5·4=100.

3) Першою цифрою може бути одна з трьох цифр: 1,3,5 (непарні).

Другою теж може бути одна з цих трьох цифр. Аналогічно і третя може бути вибрана трьома способами.

Отже, загальна кількість таких чисел 3·3·3=27.

4. В одній із версій мови БЕЙСІК ім'я змінної – це рядок з одного чи двох символів, якими можуть бути 26 букв латинського алфавіту та 10 цифр. Першим символом має бути буква. Крім того, неможна використовувати п'ять двосимвольних рядків, які зарезервовані для спеціального використання.

Скільки різних імен змінних є в цій версії мови БЕЙСІК?

Розв’язок.

Нехай V – величина, яку треба обчислити, V_1 – кількість односимвольних імен, V_2 – двосимвольних.

За правилом суми всього імен $V = V_1 + V_2$.

Очевидно V_1=26, за правилом добутку V_2 =26·36-5=931.

Отже, V=26+931=957.

Основні комбінаторні об'єкти.

Спочатку визначимо важливе поняття вибірки.

Нехай задано скінченну непорожню множину $A=\{a_1,a_2,..,a_n\}$ і виконано r таких кроків.

Крок 1. З множини A вибирають деякий елемент a_{i_1}.

Крок 2. З множини A чи $A\setminus\{a_{i_1}\}$ вибирають деякий елемент a_{i_2}.

.....

Крок r. Якщо $a_{i_1}, a_{i_2} \ldots, a_{i_{r-1}}$ – елементи, які вибрані на перших r-1 кроках $(r \geq 3)$, то на цьому кроці вибирають деякий елемент a_{i_r} з множини A або $A\setminus\bigcup_{k=1}^{r-1}\{a_{i_k}\}$. Тоді елементи $a_{i_1}, a_{i_2} \ldots, a_{i_r}$ утворюють вибірку обсягом r, або r-вибірку з множини A.

Вибірку називають впорядкованою, якщо задано порядок її елементів.

Якщо ні – невпорядкованою.

Тобто впорядкована вибірка – це кортеж (вектор) з r компонентів і позначається $(b_1, b_2, .., b_r)$, $b_i \in A$, i=1,.., r.

Невпорядковану вибірку будемо позначати $(b_1, b_2, .., b_r)$, $b_i \in A$, i=1,.., r.

Впорядковані вибірки з n-елементної множини називаються розміщеннями з n елементів по r.

Невпорядковані вибірки з n-елементної множини називаються сполученнями з n елементів по r.

Існують два способи вибору елементів.

1 спосіб. На кожному кроці вибирають елемент з усієї множини A.

Отже, один і той самий елемент з множини A може зустрітись у вибірці декілька разів.

Такі вибірки називаються вибірками з повтореннями.

2 спосіб. Вибраний елемент вилучають з множини A. Це означає, що на кожному i-му кроці $(1 < i \le k)$ вибирають елемент з множини $A \setminus \bigcup_{k=1}^{r-1} \{a_{i_k}\}$, і вибірка не містить однакових елементів.

Такі вибірки називають вибірками без повторень.

Наприклад.

Задано множину $A=\{a,b,c\}$, тобто n=3.

Наведемо розміщення без повторень з трьох елементів по два, тобто r=2: *(a,b), (a,c), (b,c), (b,a), (c,a), (c,b).*

Розміщення з повтореннями з трьох елементів по два: *(a,b), (a,c), (b,c), (b,a), (c,a), (c,b), (a,a), (b,b), (c,c).*

Сполучення без повторень з трьох елементів по два:

$[a,b]$, $[a,c]$, $[b,c]$.

Сполучення з повтореннями з трьох елементів по два:

$[a,b]$, $[a,c]$, $[b,c]$, $[a,a]$, $[b,b]$, $[c,c]$.

Тобто сполучення без повторень з n елементів по r – це просто r-елементні підмножини з n елементів, отже, їх можна записати так: $\{a,b\}$, $\{a,c\}$, $\{b,c\}$.

Сполучення з повтореннями – це не є множина у звичайному розумінні, її елементи можуть повторюватись, тобто зустрічатись більше одного разу.

Обчислення кількості розміщень і сполучень.

Кількість всіх розміщень без повторень з n елементів по r позначають A_n^r (або $A(n,r)$), де r і n - невід'ємні цілі числа, $r \le n$.

Кількість різних розміщень з повтореннями з n елементів по r позначають $\tilde{A}_n^r$ (або $\tilde{A}(n,r)$), де r і n – будь-які невід'ємні цілі числа.

Кількість всіх сполучень без повторень з n елементів по r позначають C_n^r (або $C(n,r)$), де r і n – невід'ємні цілі числа, $r \le n$.

Кількість всіх сполучень з повтореннями з n елементів по r позначають H_n^r (або $H(n,r)$), де r і n – будь-які невід'ємні цілі числа.

Числа C_n^r називаються біноміальними коефіцієнтами.

Твердження.

$$A_n^r = n(n-1)...(n-r+1) = \frac{n!}{(n-r)!} \qquad \textbf{(1)}$$

$$\tilde{A}_n^r = n^r \qquad \textbf{(2)}$$

$$C_n^r = \frac{A_n^r}{r!} = \frac{n!}{r!(n-r)!} \qquad \textbf{(3)}$$

$$H_n^r = C_{n+r-1}^r \qquad \textbf{(4)}$$

Перестановки.

Перестановка з n елементів – це особливий випадок розміщення без повторень з n елементів, коли в розміщення входять усі елементи.

Перестановки з n елементів ще називають n-перестановками.

Окремі n-перестановки різняться лише порядком елементів.

Кількість таких перестановок позначається P_n. Формулу для P_n одержують з формули (1) для кількості розміщень без повторень.

$$P_n = A_n^n = n!$$

Розглянемо задачу про перестановки з n елементів при умові, що не всі елементи різні (перестановки з повтореннями).

Точніше, нехай є n елементів k різних типів, а число $n_j (j = 1,..,k)$ – кількість елементів j-го типу.

Очевидно, що $n_1 + n_2 + .. + n_k = n$.

Перестановки з n елементів за такої умови називають перестановками з повтореннями.

Кількість таких перестановок позначається $P_n(n_1, n_2, .., n_k)$.

Щоб знайти явний вираз для $P_n(n_1, n_2, .., n_k)$, візьмемо окрему перестановку та замінимо в ній усі однакові елементи різними.

Тоді кількість різних перестановок, котрі можна отримати з узятої однієї перестановки $n_1!n_2!..n_k!$

Якщо зробити це для кожної перестановки, то одержимо $n!$ перестановок.

Отже, $P_n(n_1,n_2,..,n_k)\ n_1!n_2!..n_k! = n!$

Звідси одержуємо формулу для кількості розміщень з повтореннями:

$$P_n(n_1,n_2,..,n_k) = \frac{n!}{n_1!n_2!..n_k!}$$

Приклад 1.

Знайти кількість слів (рядків), які можна утворити, переставляючи букви слова PRODUCT.

В даному слові жодна буква не повторилася.

Отже, можна утворити $P_7 = 7! = 5040$ слів.

Приклад 2.

Знайти кількість слів, які можна утворити, переставляючи букви слова SUCCESS.

У цьому слові є повторні входження букв, тому використовують формулу для перестановок з повторенням.

$$P_7(3,2,1,1) = \frac{7!}{3!2!1!1!} = 420 \text{ слів.}$$

Отже, перестановки деяких елементів – це всі можливі способи, якими ці елементи (частіше – числа) можна вистроїти в ряд.

Підстановка – це операція, яка міняє порядок елементів в перестановці. Або можна означити по-іншому:

підстановка *n*-ї степені – це взаємно-однозначне відображення множини із *n* елементів на себе.

Для задання підстановки необхідно:

1) задати область визначення, тобто всю сукупність елементів, над якими проводиться підстановка;
2) задати алгоритм підстановки, тобто для кожного елемента з області визначення вказати елемент, в який він перейде під дією підстановки, причому різні елементи мають переходити в різні.

Введемо операцію множення множення ($\times$) над підстановками.

Добутком підстановок називається підстановка, яка одержується в результаті послідовного виконання спочатку першої, а потім другої із перемножуваних підстановок.

Наприклад, якщо $a = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_2 & x_1 & x_3 \end{pmatrix}$, $b = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_1 & x_3 & x_2 \end{pmatrix}$, то

$$a \times b = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_3 & x_1 & x_2 \end{pmatrix}.$$

Множення підстановок не є комутативним, але є асоціативним.

Одиницею групи підстановок є тотожня підстановка *I,* тобто така підстановка, кожен елемент якої переходить сам в себе

$$I = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_1 & x_2 & x_3 & x_4 \end{pmatrix}.$$

Для кожної підстановки P існує обернена P^{-1}, яка знаходиться заміною верхнього рядка на нижній. Справедливо співвідношення:

$$P \times P^{-1} = P^{-1} \times P = I.$$

Підстановки можна розкладати в добуток циклів.

Наприклад, розглянемо таку підстановку

$$P = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9 \\ 3\,5\,8\,7\,2\,0\,9\,1\,4\,6 \end{pmatrix}.$$

Запишемо її в такому вигляді:

$$P = \begin{pmatrix} 0\,3\,7\,1\,5\,2\,8\,4\,6\,9 \\ 3\,7\,1\,5\,0\,8\,4\,2\,9\,6 \end{pmatrix}.$$

Або у вигляді множення:

$P = \begin{pmatrix} 03715 \\ 37150 \end{pmatrix}\begin{pmatrix} 284 \\ 842 \end{pmatrix}\begin{pmatrix} 69 \\ 96 \end{pmatrix}$. Ці підстановки можна виконувати в довільному порядку, оскільки вони незалежні, тобто не мають спільних елементів.

Кожну зі складових підстановок можна записати у вигляді циклічної підстановки (циклу), тобто

$$P = (03715)(284)(69).$$

Будь-яку підстановку можна розкласти в добуток незалежних циклів.

Алгоритм цього розкладу: шукаємо, в який елемент перейде перший, тоді результат записуємо на 2-ге місце, шукаємо, в який елемент перейде цей 2-й елемент, записуємо на 3-тє місце, ..., продовжуємо цей процес, поки результат останньої дії не буде дорівнювати першому елементу, – це 1-й цикл. Аналогічно – інші цикли.

Цикли довжини 2 називаються транспозиціями.

Основною їх властивістю є те, що вони співпадають з оберненими до себе.

Всі підстановки можна представити у вигляді добутку транспозицій.

Наприклад, (12345) = (12)(13)(14)(15).

Або по-іншому: (12345) = (23)(24)(25)(21).

Можна дописати в розклад пару однакових транспозицій, оскільки ця пара є тотожною підстановкою (обернена до транспозиції дорівнює їй самій):

(12345) = (12)(13)(14)(15)(23)(23).

Можна в підстановку вписати елемент, який не входить в підстановку, і, переставивши елементи, виключити його:

(12345) = (01)(02)(03)(04)(05)(01).

Отже, будь-яку підстановку можна розкласти в добуток транспозицій багатьма способами, і їх в розкладі буде завжди парна або непарна кількість.

Означення 1. Якщо число транспозицій в будь-якому розкладі підстановок завжди парне, то підстановка називається парною. В противному випадку – непарною.

Означення 2. Підстановка називається парною, якщо сума інверсій її верхнього і нижнього рядків є число парне. В противному випадку – непарною.

Нехай $k_1,...,k_i,...,k_j,...,k_n$ – деяка перестановка.

Числа k_i та k_j утворюють інверсію, якщо $i<j$ і $k_i>k_j$.

Теорема. Будь-яку підстановку можна представити у вигляді добутку транспозицій сусідніх елементів.

Наслідок. Будь-яке сортування може бути виконано перестановкою сусідніх елементів.

Метод сортування, в основі якого лежить ця теорема, відомий як „метод бульбашки” (або „бульбашковий метод”).

При переміщенні елементів на своє місце транспозиціями сусідніх елементів всі елементи залишаються на своїх місцях, крім елемента, що переміщується, і того елемента, що стоїть на цільовому місці (тобто на місці, куди переміщуємо наш елемент), ці два елементи міняються місцями.

Даний метод може бути виражений у вигляді такого алгоритму:

Алгоритм 1. Сортування елементів „бульбашковим методом”

Вхід: масив *A*: array [1..*n*] of *B*, де значення елементів масиву розміщені в довільному порядку і для значень типу *B* задано відношення <.

Вихід: масив *A*: array [1..*n*] of *B*, де значення елементів масиву розміщені в порядку зростання.

```
For i from 1 to n-1 do
    m:=I {індекс кандидата в мінімальні елементи}
     for j from i+1 to n do
      if A[j]<A[m] then
```

m:=j {новий кандидат в мінімальні елементи}

end if

end for

A[j]↔A[m] {ставимо мінімальний елемент на місце}

End for

Щоб поставити мінімальний елемент на місце, використовуємо введення додаткової змінної.

II метод сортування – заміною сусідніх елементів

Даний алгоритм містить максимально *n*-1 крок. Якщо два сусідні елементи стоять „неправильно” (спочатку більший, а потім менший), то міняємо їх місцями.

Алгоритм заключається в перегляді вихідної послідовності справа наліво, і при кожному кроці менший з двох сусідніх елементів переміщується до лівої позиції (лівіше).

В результаті першого перегляду найменший елемент буде знаходитися в крайній лівій позиції.

Після цього повторюємо описаний вище процес, розглядаючи в якості вихідної послідовності масив, починаючи з другої позиції і т.д.

Алгоритм 2. Сортування методом транспозиції

```
var i,j: index, x:item;
begin
  for i:=2 to n do
    for j:=n downto i do
      if a[j-1]>a[j] then
        { x:=a[j-1];
          a[j-1]:=a[j];
          a[j]:=x
        }
end;
```

Розглянемо першу комбінаторну конфігурацію, яка ще називається задачею розкладання в ящики.

Дано *n* різних предметів і *k* ящиків. Треба розкласти в перший ящик n_1 предметів, у 2-й – n_2 предметів,.., *k*-й – n_k предметів, де $n_1+n_2+..+n_k=n$, $n_1,n_2,..,n_k$ – фіксовані числа. Скількома способами можливо це зробити?

Можна розкласти так. Серед *n* предметів візьмемо довільну n_1-підмножину і покладемо її в перший ящик (це можна зробити $C_n^{n_1}$ способами).

Серед $n-n_1$ предметів, що залишились, візьмемо n_2-підмножину й покладемо її в 2-й ящик (це можливо зробити $C_{n-n_1}^{n_2}$ способами) і продовжимо цей процес.

За правилом добутку загальна кількість розкладань дорівнює:

$$C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot .. \cdot C_{n-n_1-n_2-..-n_{k-1}}^{n_k} = \frac{n!}{n_1!(n-n_1)!} \cdot \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \cdot \ldots \cdot \frac{(n-n_1-..-n_{k-1})!}{n_k!(n-n_1-..-n_k)!} =$$

$$= \frac{n!}{n_1!n_2!..n_k!}.$$

Отже, розкладань у ящики стільки, скільки перестановок з повтореннями. Розглянемо зв'язок між цими двома задачами. Для цього занумеруємо *n* місць, які можуть займати предмети.

Кожній перестановці відповідає розподіл номерів місць на *k* класів: в *i*-й клас потрапляють номери тих місць, на які покладено предмети *i*-го типу.

Отже, знайдено відповідність між перестановками з повтореннями та розкладанням номерів місць у ящики.

Тому формули розв'язань обох задач збігаються.

Біном Ньютона.

Числа $C_n^r = \frac{n!}{r!(n-r)!}$ називається біноміальними коефіцієнтами, тобто це кількість сполучень з *n* елементів по *r*.

Зміст цієї назви випливає з теореми, яка відома як формула бінома Ньютона: $(x+y)^n = \sum_{k=0}^{n} C(n,k) x^k y^{n-k}$.

Розглянемо властивості біномних коефіцієнтів.

1. Нехай *n* і *r* – невід'ємні цілі числа, $r \le n$.

Тоді $C_n^r = C_n^{n-r}$.

Дійсно, $C_n^{n-r} = \frac{n!}{(n-r)!\left[n-(n-r)\right]!} = \frac{n!}{(n-r)!r!} = \frac{n!}{r!(n-r)!} = C_n^r$.

2. Рівність Паскаля: $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Позначимо через $S_{n,k}$ множину всіх сполучень з елементів $\{a_1,..,a_{n-1},a_n\}$ по *k* елементів.

$S_{n-1,k}$ – з елементів $\{a_1,..,a_{n-1}\}$ по *k*.

$S_{n-1,k-1}$ – з елементів $\{a_1,..,a_{n-1}\}$ по *k-1*.

Кожному сполученню з $S_{n,k}$, яке містить елемент a_n, відповідає сполучення з $S_{n-1,k-1}$.

Якщо сполучення з $S_{n,k}$ не містить a_n, то йому відповідає сполучнення з $S_{n-1,k}$.

Отже, існує бієкція між множинами $S_{n,k}$ і $S_{n-1,k} \cup S_{n-1,k-1}$.

Оскільки очевидно, що $S_{n-1,k} \cap S_{n-1,k-1} \neq \varnothing$, то $\left|S_{n,k}\right| = \left|S_{n-1,k}\right| + \left|S_{n-1,k-1}\right|$, тобто $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Отримане співвідношення дає змогу побудувати таблицю для чисел C_n^k, яке називають трикутником Паскаля.

Записують так:

1 1 n =1

1 2 1 n =2

1 3 3 1 n =3

1 4 6 4 1 n =4

……

В цьому рівнобедреному трикутнику кожне число (крім одиниць на бокових сторонах) являється сумою двох чисел, які стоять над ним.

Число сполучень C_n^r знаходиться в *(n+1)*-му рядку на *(r+1)*-му місті.

Або Трикутник Паскаля записують так:

n \ k	0	1	2	3	4	5	6	7	8	9	..
0	1										..
1	1	1									..
2	1	2	1								..
3	1	3	3	1							..
4	1	4	6	4	1						..
5	1	5	10	10	5	1					..
6	1	6	15	20	15	6	1				..
7	1	7	21	35	35	21	7	1			..
8	1	8	28	56	70	56	28	8	1		..
9	1	9	36	84	126	126	84	36	9	1	..
..	..	..	..	..	..	..	..	..	..	..	..

3. Послідовність (p_n) дійсних чисел називається унімодальною, якщо є такий унімодальний номер m, що $p_0 < p_1 < \ldots < p_m$, $p_{m+1} > p_{m+2} > \ldots > p_n$, тобто:

- послідовність строго зростає на відрізку $[0,m]$, $m>0$,

- послідовність строго спадає на відрізку $[m+1,n]$, $m+1<n$,

- максимальне значення досягається не більше, ніж у двох точках: m і, можливо, $m+1$.

Через $\lfloor x \rfloor$ позначимо найбільше ціле число, яке менше або дорівнює x (цілу частину числа x).

Наприклад, $\lfloor 3{,}14 \rfloor = 3$, $\lfloor -3{,}14 \rfloor = -4$.

Теорема. При фіксованому n послідовність біноміальних коефіцієнтів (C_n^k), k=0,1,..,n, унімодальна, $m = \left\lfloor \frac{n}{2} \right\rfloor$.

Якщо n – парне, то максимум досягається в точці $m = \left\lfloor \frac{n}{2} \right\rfloor = \frac{n}{2}$, якщо непарне, то – у двох точках $m = \left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$ і $m+1 = \frac{n+1}{2}$.

4. Рівність Вандермонда.

Нехай m,n,r – невід'ємні цілі числа, причому $r \le \min\{m,n\}$. Тоді $C_{n+m}^r = \sum_{k=0}^{r} C_m^{r-k} C_n^k$.

Теорема біноміальна.

Нехай x та y – змінні, n – додатне ціле число.

Тоді $(x+y)^n = \sum_{j=0}^{n} C_n^j x^j y^{n-j} = \sum_{j=0}^{n} C_n^j x^{n-j} y^j$.

Цю рівність легко отримати.

Оскільки $x^j y^{n-j}$ одержано внаслідок j-кратного вибору x і $(n-j)$- кратного вибору y з n співмножників у виразі $(x+y)^n$, то коефіцієнт при $x^j y^{n-j}$ дорівнює кількості способів j-кратного вибору x з n співмножників, тобто C_n^j.

Друга рівність випливає з того, що $C_n^j = C_n^{n-j}$.

Легко перевірити, що $(x-y)^n = \sum_{j=0}^{n} (-1)^j C_n^j x^{n-j} y^j$.

Приклад 1.

Знайти розклад виразу $(x+y)^4$.

За біноміальною теоремою одержимо:

$$(x+y)^4 = C_4^0 x^4 + C_4^1 x^3 y + C_4^2 x^2 y^2 + C_4^3 x y^3 + C_4^4 y^4 = x^4 + 4x^3 y + 6x^2 y^2 + 4x y^3 + y^4.$$

Біноміальні коефіцієнти можна брати з трикутника Паскаля або обчислювати за формулою (3) (за формулою для кількості сполучень).

Приклад 2.

Визначити коефіцієнт при $x^{12} y^{13}$ в розкладі $(x+y)^{25}$.

Очевидно, цей коефіцієнт визначається за формулою

$$C_{25}^{13} = \frac{25!}{13!12!} = 5200300 .$$

За допомогою біноміальної теореми можна одержати ще дві властивості біноміальних коефіцієнтів:

5. $\sum_{k=0}^{n} C_n^k = 2^n .$

Дійсно, $2^n = (1+1)^n = \sum_{k=0}^{n} C_n^k 1^{n-k} 1^k = \sum_{k=0}^{n} C_n^k .$

Або по-іншому можна показати цю рівність. Оскільки C_n^k – число різних *k*-елементних підмножин *n*-елементної множини, то сума всіх таких чисел становить число всіх підмножин даної *n*-елементної множини.

6. $\sum_{k=0}^{n} (-1)^k C_n^k = 0 .$

Аналогічно $0 = [1+(-1)]^n = \sum_{k=0}^{n} C_n^k 1^{n-k} (-1)^k = \sum_{k=0}^{n} (-1)^k C_n^k .$

Поліноміальна теорема.

Як узагальнення бінома розглянемо вираз

$$(x_1 + x_2 + ... + x_k)^n .$$

Теорема.

Вираз $(x_1 + x_2 + ... + x_k)^n$ дорівнює сумі всіх можливих доданків $P_n(n_1, n_2, .., n_k) x_1^{n_1} x_2^{n_2} ... x_k^{n_k}$, де $n_1 + n_2 + ... + n_k = n$, тобто

$$(x_1 + x_2 + ... + x_k)^n = \sum_{\substack{n_1 \geq 0, ..., n_k \geq 0 \\ n_1 + ... + n_k = n}} P_n(n_1, .., n_k) x_1^{n_1} x_2^{n_2} ... x_k^{n_k}.$$

Покажемо це. Запишемо $(x_1 + x_2 + ... + x_k)^n$ у вигляді добутку n співмножників і розкриємо дужки.

Коефіцієнт при $x_1^{n_1} x_2^{n_2} ... x_k^{n_k}$ дорівнює кількості перестановок із повтореннями, таких, що елемент x_1 міститься в кожній з них n_1 разів, x_2 – n_2 разів,..., x_k входить n_k разів, а всього елементів $n_1 + n_2 + ... + n_k = n$. Очевидно, що цей коефіцієнт дорівнює $P_n(n_1, .., n_k)$.

Отриману формулу називають поліноміальною.

Вона дає змогу доводити деякі властивості чисел $P_n(n_1, .., n_k)$. Зазначимо дві з них.

1. Нехай $x_1 = x_2 = ... = x_k = 1$, тоді $\sum_{\substack{n_1 \geq 0, ..., n_k \geq 0 \\ n_1 + ... + n_k = n}} P_n(n_1, n_2 .., n_k) = k^n$.

2. Помножимо обидві частини поліноміальної формули для $n-1$ на $(x_1 + x_2 + ... + x_k)$ та порівняємо коефіцієнти при однакових доданках. Одержимо таке співвідношення:

$$P_n(n_1, n_2, .., n_k) = P_{n-1}(n-1, n_2, .., n_k) + P_{n-1}(n_1, n_2 - 1, .., n_k) + ... + P_{n-1}(n_1, n_2, .., n_k - 1).$$

Задача про цілочислові розв'язки.

Формулюється так:

Знайти кількість розв’язків рівняння $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах, де n – ціле , невід'ємне число.

Взявши такі невід'ємні числа $x_1, x_2, ..., x_r$, що $x_1 + x_2 + ... + x_r = n$, можна одержати сполучення з повтореннями з r елементів по n, а саме: елементів першого типу – x_1 одиниць, другого – x_2,..., r-го – x_r.

Навпаки, якщо є сполучення з повтореннями з r елементів по n, то кількості елементів кожного типу задовольняють рівнянню $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах.

Отже, кількість цілих невід'ємних розв'язків цього рівняння $H_r^n = C_{r+n-1}^n = \frac{(r+n-1)!}{n!(r-1)!}$.

Приклад 1

Знайти кількість невід'ємних цілих розв'язків рівняння $x_1 + x_2 + x_3 = 11$.

За формулою: $H_3^{11} = C_{3+11-1}^{11} = C_{13}^{11} = \frac{13!}{11!2!} = \frac{12 \cdot 13}{2} = 78$.

Кількість розв'язків рівняння $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах можна визначити і тоді, коли на ці змінні накладено певні обмеження.

Приклад 2. Знайти кількість невід'ємних цілих розв'язків рівняння $x_1 + x_2 + x_3 = 11$ за умов $x_1 \geq 1,\ x_2 \geq 2,\ x_3 \geq 3$.

Очевидно, що ця задача еквівалентна рівнянню $x_1 + x_2 + x_3 = 5$ без обмежень.

Дійсно, треба взяти щонайменше один елемент першого типу, два елементи другого типу, три елементи третього – разом 1+2+3=6 елементів.

Отже, 11-5=6 елементів залишається для довільного вибору.

$$H_3^5 = C_{4+11-1}^{11} = C_{14}^{11} = \frac{14!}{11!3!} = 364.$$

Властивості біноміальних коефіцієнтів.

1. Формула симетрії $C_{n+m}^{n} = C_{n+m}^{m}$.

2. Формула додавання $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

3. Формула суми всіх біноміальних коефіцієнтів

$C_0^0 - C_n^1 + C_n^2 - ... + (-1)^n C_n^n = 0$.

4. Формула винесення за дужки $C_n^k = \frac{n}{k} \cdot C_{n-1}^{k-1}$.

5. $C_n^0 + C_{n-1}^1 + ... + C_{n+k}^n = C_{n+k+1}^n$.

6. $C_{n-1}^{r-1} + C_{n-2}^{r-1} + ... + C_{r-1}^{r-1} = C_n^r$.

7. $C_n^0 + 2C_n^1 + ... + nC_n^n = n2^{n-1}$.

8. $C_n^0 C_m^k + C_n^1 C_m^{k-1} + ... + C_n^k C_m^0 = C_{n+m}^k$.

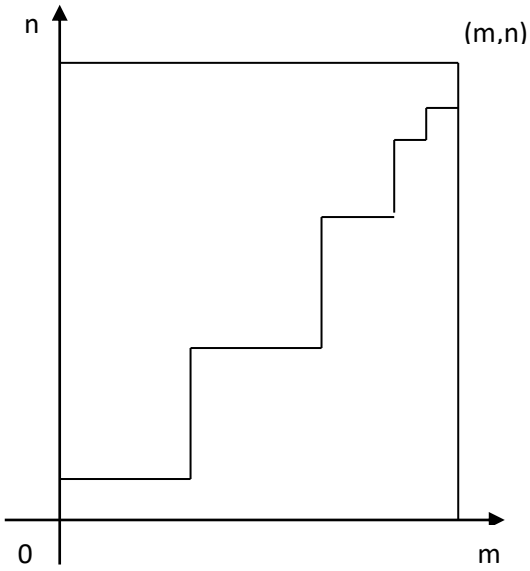
9. $C_n^m C_k^0 + C_{n-1}^{m-1} C_{k+1}^1 + ... + C_{n-m}^0 C_{k+m}^m = C_{n+k+1}^m$.

10. $(C_n^0)^2 + (C_n^1)^2 + ... + (C_n^n)^2 = C_{2n}^n$.

Це не всі властивості біноміальних коефіцієнтів. Є велика їх кількість і виявляються все нові і нові.

Геометрична інтерпретація біноміальних коефіцієнтів.

Нехай маємо прямокутну шахову дошку *m* на *n*, яка розміщена на координатній площині.



Ця дошка складається з *m-n* елементарних квадратів, розділених *n-1* горизонтальною і *m-1* вертикальною лініями.

Визначимо, скількома різними найкоротшими шляхами можна потрапити з точки (*0,0*) в точку (*m,n*) на цій дошці.

Кожен найкоротший шлях, який веде з точки (*0,0*) в точку (*m,n*), складається з *m+n* сторін елементарних квадратів, серед яких *m* горизонтальних і *n* вертикальних.

Ці шляхи розрізняються лише числом вертикальних і горизонтальних сторін.

Отже, загальна кількість шляхів дорівнює числу способів, якими з *m+n* сторін можна вибрати *n* вертикальних, тобто це число рівно C_{m+n}^{n}.

Можна було підрахувати не за вертикальними сторонами, а за горизонтальними.

Тобто існує C_{m+n}^{m} різних найкоротших шляхів, і отже, справджується рівність $C_{m+n}^{n} = C_{m+n}^{m}$ – формула симетрії.

Приклади.

1. Збірна команда університету з волейболу налічує 15 чоловік.

Скільки різних варіантів повинен розглянути тренер перед грою, щоб заявити список гравців на гру?

Розв'язок.

Кількість гравців волейбольної команди 6. Отже, число всіх можливих варіантів – це число різних підмножин, які складаються з шести елементів у множині з 15-ти елементів, тобто

$$C_{15}^{6} = \frac{15!}{6!(15-6)!} = \frac{15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 5 \cdot 7 \cdot 13 \cdot 11 = 5005.$$

2. У скількох точках перетинаються діагоналі випуклого десятикутника, якщо жодні три з них не перетинаються в одній точці?

Розв'язок.

Кожній точці перетину двох різних діагоналей відповідають чотири вершини десятикутника, а кожним чотирьом вершинам – одна точка перетину.

Таким чином, число всіх точок перетину дорівнює числу способів, якими з 10-ти вершин можна вибрати чотири вершини, тобто

$$C_{10}^{4} = \frac{10!}{4!6!} = \frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2} = 210.$$

3. В деяких видах спортивних змагань результатом являється визначення учасників, які зайняли 1-ше, 2-ге і 3-тє місця.

Скільки існує різних результатів, якщо в змаганні бере участь *n* учасників?

Розв'язок.

Кожен можливий результат відповідає функції $F:\{1,2,3\}\to\{1...n\}$ (аргумент – номер призового місця, результат – номер учасника).

Таким чином, всього можливо $A_n^3 = n(n-1)(n-2)$ різних результатів.

4.На початку гри в доміно кожному учаснику гри видається 7 костів з 28 різних, що є в наявності.

Скільки існує різних комбінацій костів, які учасник може отримати на початку гри?

Розв'язок.

Потрібна кількість дорівнює кількості 7-елементних підмножин 28-елементної множини :

$$C_{28}^7 = \frac{28!}{7!(28-7)!} = \frac{28\cdot 27\cdot 26\cdot 25\cdot 24\cdot 23\cdot 22}{7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1} = 1184040.$$

5.Скількома способами можна розсадити *n* нових гостей серед *m* гостей, які вже сидять за круглим столом?

Розв'язок.

Між *m* гостями, що вже сидять існує *m* проміжків, в які можна розсадити нових. Тобто, це можна зробити

$$H_m^n = C_{m+n-1}^n = \frac{(m+n-1)!}{n!(m-1)!}$$ способами.

Генерація перестановок.

Проблемі систематичної побудови всіх *n!* перестановок *n*-елементної множини присвячено багато публікацій.

Ця проблема має давню історію.

Її появу можна віднести до початку XVII ст., коли в Англії виникло особливе мистецтво дзвонарства. Воно полягало у вибивані на *n* різних дзвонах усіх *n!* перестановок. Це треба було робити по пам'яті.

Тому шанувальники цього мистецтва розробили перші прості методи систематичної побудови всіх перестановок без повторень.

Деякі з цих методів було знову відкрито в наш час у зв'язку з появою комп'ютерів.

“Книга рекордів Гіннеса” містить інформацію про вибивання всіх 8!=40320 перестановок на восьми дзвонах у 1963р.

Для цього було потрібно 17 год. 58 хв. 30 с.

З використанням комп'ютерів генерування перестановок здійснюється дуже швидко.

Кожній *n*-елементній множині *A* можна поставити у взаємно-однозначну відповідність множину $A' = \{1,2,..,n\}$.

Зручно спочатку генерувати перестановки *n* перших натуральних чисел, а потім замінити кожне число відповідним елементом множини *A*.

Внаслідок цього отримаємо всі перестановки елементів деякої множини *A*.

Генерування перестановок в лексикографічному порядку.

Існують різні алгоритми побудови всіх перестановок множини $A' = \{1,2,..,n\}$.

Розглянемо один з них.

Цей алгоритм ґрунтується на послідовній побудові перестановок множини A' у лексикографічному порядку.

Перестановку $(a_1, a_2,.., a_n)$ будемо позначати $a_1 a_2 .. a_n$.

На множині всіх перестановок (в загальному – на множині всіх кортежів довжиною n з елементами з множини $A' = \{1,2,...,n\}$) визначимо лексикографічний порядок: $a_1 a_2 ... a_h < b_1 b_2 ... b_h$,

якщо для деякого k, $1 \le k \le n$, виконуються співвідношення $a_1 = b_1$, $a_2 = b_2, \ldots, a_{k-1} = b_{k-1}$ але $a_k < b_k$.

В цьому випадку кажуть, що перестановка $a_1 a_2 ... a_n$ менше від перестановки $b_1 b_2 ... b_n$ або перестановка $b_1 b_2 ... b_n$ більше від перестановки $a_1 a_2 ... a_n$.

Якщо замість чисел $1,2,\ldots,n$ взяти букви $a,b,\ldots,z$ з природним порядком $a < b < ... < z$, то лексикографічний порядок – стандартна послідовність, у якій слова довжиною n наведено в словнику.

Перестановку $b_1 b_2 ... b_n$ називають лексикографічно наступною за $a_1 a_2 ... a_n$, якщо не існує такої перестановки $c_1 c_2 ... c_n$, що $a_1 a_2 ... a_n < c_1 c_2 ... c_n$ і $c_1 c_2 ... c_n < b_1 b_2 ... b_n$.

Приклад 1.

Перестановка 23415 множини $\{1,2,3,4,5\}$ менша від перестановки 23514.

Алгоритм генерування перестановок множини $A' = \{1,2,...,n\}$ ґрунтується на процедурі, що будує перестановку, лексикографічно наступну за даною перестановкою $a_1 a_2 .. a_n$.

Покажемо, як це можна зробити.

Спочатку припустимо, що $a_{n-1} < a_n$.

Поміняємо місцями a_{n-1} і a_n.

Одержимо більшу перестановку.

Вона лексикографічно наступна, бо жодна інша підстановка не більша за дану перестановку й не менше за отриману.

Приклад 2.

Нехай 234156 – задана перестановка, тоді перестановка 234165 лексикографічно наступна.

Розглянемо випадок, коли $a_{n-1} > a_n$. Переглянемо останні три члени перестановки.

Якщо $a_{n-2} < a_{n-1}$, то останні три члени можна переставити для отримання наступної перестановки.

Поставимо менше з двох чисел a_{n-1} і a_n, яке, однак, більше, ніж a_{n-2},

на позицію *n*-2.

Потім розмістимо число, яке залишилося, і a_{n-2} на останніх двох позиціях у висхідному порядку.

Приклад.3.

Нехай 234165 – задана перестановка і тоді перестановка 234516 лексикографічно наступна.

Отже, маємо такий алгоритм.

Алгоритм побудови лексикографічно наступної перестановки за перестановкою $a_1 a_2 .. a_n$ **.**

Кроки алгоритму:

Крок 1. Знайти такі числа a_j і a_{j+1}, що $(a_j < a_{j+1}) \wedge (a_{j+1} > a_{j+2} > ... > a_n)$.

Для цього треба знайти в перестановці першу справа пару сусідніх чисел, у якій число, що ліворуч, менше від числа, що праворуч.

Крок 2. Записати в *j*-ту позицію таке найменше з чисел $a_{j+1}, a_{j+2}, ..., a_n$, яке водночас більше, ніж a_j.

Крок 3. Записати у висхідному порядку число a_j і решту чисел $a_{j+1}, a_{j+2}, ..., a_n$ у позиції $j+1, ..., n$.

Обґрунтування.

Покажемо, що не існує перестановки, яка водночас більша від $a_1a_2..a_n$, але менша від побудованої за цим алгоритмом.

Це означає, що побудована перестановка дійсно лексикографічно наступна за даною перестановкою $a_1a_2..a_n$.

Справді, за наведеним алгоритмом нова перестановка збігається зі старою в позиціях 1,…,*j*-1.

У *j*-й позиції нова перестановка містить a_k, а стара – a_j, причому $a_k > a_j$.

Отже, нова перестановка лексикографічно більша від старої.

Крім того, вона перша в лексикографічному порядку з $a_1, a_2, .., a_{j-1}, a_k$ у позиціях з 1 до *j*.

Стара перестановка остання з $a_1, a_2, .., a_{j-1}, a_j$ у цих самих позиціях.

Згідно з алгоритмом a_k вибирають найменшим з $a_{j+1}, a_{j+2}, ..., a_n$, але більшим, ніж a_j.

Отже, не існує жодної перестановки між старою та новою.

Приклад. 4.

Побудуємо наступну в лексикографічному порядку за перестановкою 362541.

Остання пара чисел, у якій перше число менше за друге, – 25.

Отже, розглянемо послідовність чисел 541.

Серед них найменше число, більше від 2, це – 4.

Тепер 4 запишемо на місце 2, а решту чисел 251 розмістимо на останніх трьох позиціях у висхідному порядку: 364125.

Щоб побудувати всі $n!$ перестановок множини $A' = \{1,2,...,n\}$, починаємо з лексикографічної найменшої перестановки $123...n$ і послідовно $n!$-1 разів виконуємо алгоритм побудови лексикографічно наступної перестановки.

$n!=4!=1\cdot2\cdot3\cdot4=24 \rightarrow$ 24 перестановки.

1)	1234	11)	2413	21)	4213
2)	1243	12)	2431	22)	4231
3)	1324	13)	3124	23)	4312
4)	1342	14)	3142	24)	4321
5)	1423	15)	3214		
6)	1432	16)	3241		
7)	2134	17)	3412		
8)	2143	18)	3421		
9)	2314	19)	4123		
10)	2341	20)	4132		

Генерування перестановок в антилексикографічному порядку.

Інакше кажучи, на множині перестановок природнім чином можна визначити впорядкованість на основі впорядкованості елементів.

Тобто:

Кажуть, що перестановка $< a_1,...,a_n >$ лексикографічно передує перестановці $< b_1,...,b_n >$, якщо $\exists k \le n \quad (a_k < b_k) \wedge (\forall i < k \ ) \quad a_i = b_i$.

Кажуть, що перестановка $< a_1,...,a_n >$ антилексикографічно передує перестановці $< b_1,...,b_n >$, якщо $\exists k \le n \quad (a_k > b_k) \wedge (\forall i > k) \quad a_i = b_i$.

Можна визначити так: кажуть, що перестановка $a_1a_2...a_n$ антилексикографічно передує перестановці $в_1в_2...в_n$, якщо для деякого k виконуються співвідношення:

$a_k > b_k$, $a_{k+1} = b_{k+1}$, ..., $a_n = b_n$.

Наприклад,

(1,2,3) антилексикографічно передує (2,1,3);

(2,1,3) антилексикографічно передує (1,3,2);

(1,3,2) антилексикографічно передує (3,1,2);

(3,1,2) антилексикографічно передує (2,3,1);

(2,3,1) антилексикографічно передує (3,2,1).

Розглянемо алгоритм, який генерує всі перестановки елементів $1,\ldots,n$ в антилексикографічному порядку.

Масив *P*: *array* [1..*n*] *of* [1..*n*] є глобальним і призначений для зберігання перестановок.

Алгоритм 2. Генерація перестановок в антилексикографічному порядку

Вхід: *n* – кількість елементів,

Вихід: послідовність перестановок елементів $1,\ldots,n$ в антилексикографічному порядку.

for i from 1 *to n do*

$P[i] := 1\{\text{ініціалізація}\}$

end for

Antilex (n) {виклик рекурсивної процедури *Antilex*}

Основна робота по генерації перестановок виконується рекурсивною процедурою *Antilex*.

Вхід: *m* – параметр процедури – кількість перших елементів масиву *P*,

для яких генеруються перестановки.

Вихід: послідовність перестановок $1,\ldots,m$ в антилексикографічному порядку.

```
if m=1 then
  yield P {чергова перестановка}
else
  for i from 1 to m do
    Antilex (m-1) {рекурсивний виклик}
    if  i<m then
      P[i] ↔ P[m] {наступний елемент}
      Reverse (m-1) {зміна порядку елементів}
    end if
  end for
end if
```

Допоміжна процедура *Reverse* переставляє елементи заданого відрізку масиву *P* в оберненому порядку.

Вхід: *k* – номер елемента, що задає відрізок масиву *P*, який має бути переставлений в оберненому порядку.

Вихід: перші *k* елементів масиву *P*, переставлені в оберненому порядку

$j := 1$ {нижня границя діапазону обернення}.

while $j<k$ *do*

$P[j] \leftrightarrow P[k]$

$j := j+1$

$k := k-1$

end while

Обґрунтування:

Шукану послідовність перестановок *n* елементів можна отримати з послідовності перестановок *n*-1 елемента таким чином.

Треба виписати *n* блоків по (*n*-1)! перестановок в кожному, що відповідають послідовності перестановок (*n*-1) елемента в анти-лексикографічному порядку. Потім до всіх перестановок в першому блоці треба приписати справа *n*, у другому (*n*-1) і аналогічно в інших у спадному порядку.

Тоді в кожному з блоків (крім першого), до перестановок якого справа приписаний елемент *i*, потрібно в перестановках блоку замінити всі входження елемента *i* на елемент *n*.

В одержаній послідовності всі перестановки різні, і їх *n*(*n*-1)!=*n*!, тобто перераховані всі перестановки.

При цьому антилексикографічний порядок збережений для послідовностей всередині одного блоку, бо цей порядок був збережений у вихідній послідовності, а для послідовностей на границях двох блоків, тому що відбувається зменшення крайнього правого елемента.

Звернемося до процедури *Antilex* – легко бачити, що в ній реалізована вказана побудова.

В основному циклі спочатку будується черговий блок – послідовність перестановок перших m-1 елементів масиву P (при цьому елементи $P[m],..,P[n]$ залишаються незмінними).

Потім елемент $P[m]$ міняється місцями з черговим елементом $P[i]$.

Виклик допоміжної процедури *Reverse* необхідний, оскільки остання перестановка в блоці являється оберненням першої, а для генерації наступного блоку на черговому кроці циклу потрібно відновити вихідний порядок.

Приклад.

Послідовність перестановок в антилексикографічному порядку для $n=3$: *(1,2,3), (2,1,3), (1,3,2), (3,1,2), (2,3,1), (3,2,1).*

Генерування підмножин.

Елементи множини $\{1,..,m\}$ впорядковані. Тому кожна *n*-елементна підмножина теж може бути розглянута як впорядкована послідовність.

На множині таких послідовностей природнім чином визначається лексикографічний порядок.

Розглянемо алгоритм, що генерує всі *n*-елементні підмножини *m*-елементної множини в лексикографічному порядку.

Алгоритм 3. Генерація *n*-елементних підмножин *m*-елементної множини.

Вхід: *n* – потужність підмножини, *m* – потужність множини, $m \geq n > 0$.

Вихід: послідовність всіх *n*-елементних підмножин *m*-елементної множини в лексикографічному порядку.

for I from 1 *to m do*

$A[i] := i$ {ініціалізація вихідної множини}

end for

if m=n then

yield $A[1..n]$ {єдина підмножина}

exit

end if

p:=n {*p* – номер першого елемента, що змінюється}

while $p \geq 1$ *do*

yield $A[1..n]$ {чергова підмножина в перших *n* елементах масиву *A*}

if $A[n] = m$ *then*

p:=p -1 {неможливо збільшити останній елемент}

else

p:=n {можна збільшити останній елемент}

end if

if $p \geq 1$ *then*

for i from n dowhto p do

$A[i] := A[p] + i - p + 1$ {збільшення елементів}

end for

end if

end while.

Обґрунтування.

У вихідній послідовності n-елементних підмножин (кожна з яких являється зростаючою послідовністю n чисел із діапазона 1..m) за послідовністю $< a_1,..,a_n >$ слідує послідовність

$$< b_1,..,b_n >=< a_1,..,a_{p-1}, a_p+1, a_p+2,..,a_p+n-p+1 >,$$

де p – максимальний індекс, для якого $b_n = a_p + n - p + 1 \le m$.

Іншими словами, наступна послідовність одержується із попередньої заміною деякої кількості елементів у “хвості” послідовності на цілі числа, що слідують підряд один за одним, але так, щоб останній елемент не перевищував m, а перший елемент, що змінюється, був на 1 більше, ніж відповідний елемент в попередній послідовності.

Таким чином, індекс p, починаючи з якого потрібно змінити “хвіст послідовності”, визначається за значенням елемента $A[n]$.

Якщо $A[n] < m$, то потрібно змінити лише $A[n]$, і при цьому $p:=n$.

Якщо вже $A[n] = m$, то потрібно зменшувати індекс $p:=p$ -1, збільшуючи довжину “хвоста”, що змінюється.

Приклад.

Послідовність n-елементних підмножин m-елементної множини в лексикографічному порядку для n=3 і m=4:

(1,2,3), (1,2,4), (1,3,4), (2,3,4).

Генерування сполучень

Знову будемо розглядати, множину $A=\{1,2,\ldots,n\}$.

Генерування сполучень аналогічно генеруванню підмножин.

Розглянемо алгоритм побудови лексикографічно наступного сполучення.

Алгоритм побудови лексикографічно наступного сполучення.

Кроки алгоритму:

Крок 1.Знайти в рядку перший справа елемент a_i такий, що $a_i \neq n-r+i$.

Крок 2.Для знайденого елемента виконати присвоєння $a_i := a_i + 1$. .

Крок 3.Для $j=i+1, i+2,..., r$ виконати: $a_j:=a_i+j-i$ (або, що те саме, $a_j=a_{j-1}+1$).

Приклад.

Нехай $A' = \{1,2,3,4,5,6\}$. Знайти сполучення, наступне за {1,2,5,6} в лексикографічному порядку.

Подаємо це сполучення рядком 1256, маємо $n=6$, $r=4$.

Перший справа з таких елементів, що $a_i \neq 6-4+i$, це $a_2=2$. .

Для обчислення наступного більшого сполучення збільшуємо a_2 на 1 і одержуємо $a_2=3$. Тепер нехай $a_3=3+1=4$ і $a_4=3+2=5$.

Отже, наступне в лексикографічному порядку сполучення – те, що зображене рядком 1345, тобто {1,3,4,5}.

Лексикографічний порядок – такий, як і для перестановок:

$b_1 b_2 ... b_n$ наступне за $a_1 a_2 ... a_n$, якщо $\exists k \le n \quad b_k > a_k, \forall i < k \quad a_i = b_i$.

Це визначення можливе, оскільки сполучення – це підмножина і порядок запису елементів множини є неістотний, тому будемо записувати елементи в кожному сполученні у висхідному порядку, наприклад

{3,5,1} будемо записувати {1,3,5}.

Отже, сполучення $\{a_1,a_2,...,a_r\}$ розглядатимемо як рядок чисел $a_1a_2...a_r$, причому $a_1<a_2<...a_r$.

Розглянемо лексикографічний порядок.

Нехай n=5, r=3

Якщо можна збільшити останню цифру, то так і будемо робити.

Тому маючи рядок 123, його можна замінити на 124.

Якщо ж маємо 125, останнє число збільшити не можна.

Тому переходимо до наступного (справа) числа і дивимось, чи можна його збільшити.

У даному випадку це можна зробити: потрібно замінити 2 на 3.

Проте ми прагнемо побудувати найменший рядок із тих, які більші від 125.

Тому збільшуємо останнє число (тобто 3) на 1 і записуємо результат у наступну позицію.

Отже, перші два числа – 1 і 3, тому наступний рядок – 134.

Припустимо, що є рядок 145. Останнє і передостаннє числа збільшити неможна.

Проте перше число можна збільшити, тому замість 1 пишемо 2.

Щоб зробити рядок мінімальним, в якості останніх чисел візьмемо 3 та 4, отже, отримаємо рядок 234.

Узагальнюємо.

Значення останнього числа в рядку – максимально можливе, якщо воно дорівнює $n=n-r+r$.

Якщо останнє число – максимально можливе, то передостаннє – максимально можливе, якщо воно дорівнює $n-r+(r-1)$ або $n-r+i$, де $i=r-1$ – позиція цього числа.

Тобто, значення кожного i–го числа максимально можливе, якщо числа праворуч від нього – максимально можливі, і це значення дорівнює $n-r+i$.

Отже, проглядаємо рядок справа наліво і визначаємо, чи дорівнює значення i- го елемента $n-r+i$ (це максимальне значення, яке може бути в i-й позиції).

Перше значення, яке не задовольняє цю умову, можна збільшити.

Нехай, наприклад, це значення дорівнює m і займає j-ту позицію. Збільшуємо m на 1, а значення кожного елемента, що стоїть після j-го, дорівнює значенню попереднього елемента плюс 1.

Приклади.

1.Обчислити значення:

$A_5^3; A_6^5, A_8^1, A_8^5, A_8^8, A_{10}^9, A_{n+4}^{n-2}$.

$\boxed{A_n^r = \frac{n!}{(n-r)!}}$ – кількість розміщень без повторень з n елементів по r .

$$A_5^3 = \frac{5!}{(5-3)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5}{2!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5}{1\cdot 2} = 3\cdot 4\cdot 5 = 60$$

$$A_6^5 = \frac{6!}{(6-5)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6}{1!} = 6\cdot 20\cdot 6 = 36\cdot 20 = 720$$

$$A_8^1 = \frac{8!}{(8-1)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7} = 8$$

$$A_8^5 = \frac{8!}{(8-5)!} = \frac{8!}{3!} = \frac{3!\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8}{3!} = 20\cdot 8\cdot 42 = 160\cdot 42 = 6720$$

$$A_8^8 = \frac{8!}{0!} = 40320$$

$$A_{10}^9 = \frac{10!}{(10-9)!} = \frac{10!}{1} = 10!$$

2.Обчислити значення:

$C_5^1; C_5^3, C_8^4, C_{12}^6$

$\boxed{C_n^r = \frac{n!}{r!(n-r)!}}$ – кількість розміщень без повторень з n елементів по r .

$$C_5^1 = \frac{5!}{1!(5-1)!} = \frac{5!}{4!} = \frac{4*!5}{4!} = 5$$

$$C_5^3 = \frac{5!}{3!(5-3)!} = \frac{3!\cdot 4\cdot 5}{3!\cdot 2!} = \frac{20}{2} = 10$$

$$C_8^4 = \frac{8!}{4!(8-4)!} = \frac{4\cdot!5\cdot 6\cdot 7\cdot 8}{4!\cdot 4!} = \frac{10\cdot 7\cdot 2}{2} = 70$$

$$C_{12}^6 = \frac{12!}{6!(12-6)!} = \frac{6!\cdot 7\cdot 8\cdot 9\cdot 10\cdot 11\cdot 12}{6!6!} = \frac{7\cdot 8\cdot 9\cdot 10\cdot 11\cdot 12}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6} = \frac{7\cdot 8\cdot 9\cdot 11}{3\cdot 2} = 7\cdot 4\cdot 3\cdot 11 = 924$$

3.Нехай M=$\{1,2,3,4,5\}$.Навести всі розміщення та сполучення без повторень з елементів множини M по 3 елементи.

4.Скількома способами можна визначити призові місця (1-ше, 2-ге, 3-тє) у забігу 12 коней?

$$F\{1,2,3\} \to \{1.....12\}$$

$$\mathrm{A}_{12}^3 = \frac{12!}{(12-3)!} = \frac{12!}{9!} = \frac{9!\cdot 10\cdot 11\cdot 12}{9!} = 110\cdot 12 = 1320\,.$$

5.Скільки різних слів (рядків) можна утворити зі слова MISSISSIPPI, використовуючи всі букви?

В цьому слові є повторні входження букв, тому використовують. формулу для перестановок з повтореннями

$$P_n(n_1, n_2, ..., n_k) = \frac{n!}{n_1! n_2! ..... n_k!}$$

$$P_{11}(4,4,1,2) = \frac{11!}{4! \cdot 4! \cdot 1! \cdot 2!} = \frac{4! \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11}{4! \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 2} = 5 \cdot 7 \cdot 9 \cdot 10 \cdot 11 = 35 \cdot 9 \cdot 110 = 34650.$$

6.Знайти лексикографічно наступну перестановку для кожної з перестановок

1432; 54123; 12453; 45231; 6714235; 31528764.

$\underbrace{1}\underbrace{4}32$ $54\underbrace{1}\underbrace{2}3$ $12\underbrace{4}\underbrace{5}3$ $45\underbrace{2}\underbrace{3}1$ $6714\underbrace{2}\underbrace{3}5$ $315\underbrace{2}\underbrace{8}764$

2134 54132 12534 45312 6714253 31542678.

7.Розмістити наведені перестановки елементів множини $\{1,2,3,4,5,6\}$ у лексикографічному порядку

234561, 231456, 165432, 156423, 543216,

541236, 231465, 314562, 432561, 654321,

654312, 435612.

Відповідь: 1) 156423; 2) 165432; 3) 231456; 4) 231465; 5) 234561; 6)314562; 7) 432561; 8) 435612; 9) 541236; 10) 543216; 11) 654312; 12) 654321

8.За допомогою алгоритму побудови лексикографічно наступної перестановки записати перші 132 перестановки елементів множини $\{1,2,3,4,5,6\}$.

123456	126435	135624	145236	153426
123465	126453	135642	145263	153462
123546	126534	136245	145326	153624
123564	126543	136254	145362	153642
123645	132456	136425	145623	154236
123654	132465	136452	145632	154263
124356	132546	136524	146235	154326
124365	132564	136542	146253	154362
124536	132645	142356	146325	154623
124563	132654	142365	146352	154632

124635 134256 142536 146523 156234

124653 134265 142563 146532 156243

125346 134526 142635 152346 156324

125364 134562 142653 152364 156342

125436 134625 143256 152436 156423

125463 134652 143265 152463 156432

125634 135246 143526 152634 162345

125643 135264 143562 152643 162354

126345 135426 143625 153246 162435

126354 135462 143652 153264 162453

162534 164523 213645

162543 164532 213654

163245 165234 214356

163254 165243 214365

163425 165324 214536

163452 165342 214563

163524 165423 214635

163542 165432 214653

164235 213456

164253 213465

164325 213546

164352 213564

9.Задати взаємно-однозначну відповідність між елементами множини M={*a,b,c,d,e*} і X={1,2,3,4,5}.

Побудувати перші 28 перестановок елементів множини M у лексикографічному порядку.

Задамо взаємно-однозначну відповідність між елементами даних множин таким чином:

$a \Leftrightarrow 1$

$b \Leftrightarrow 2$

$c \Leftrightarrow 3$

$d \Leftrightarrow 4$

$e \Leftrightarrow 5$

$\underbrace{12345}$

abcde

$\underbrace{12354}$

abced

$\underbrace{12435}$

abdce

$\underbrace{12453}$

abdec

$\underbrace{12534}$

abecd

$\underbrace{12543}$

abedc

$\underbrace{13245}$

acbde

$\underbrace{13254}$

acbed

$\underbrace{13425}$

acdbe

$\underbrace{13452}$

acdeb

$\underbrace{13524}$

acebd

$\underbrace{13542}$

acedb

14235 *adbce*

14253 *adbec*

14325 *adcbe*

14352 *adceb*

14523 *adebc*

14532 *adecb*

15234 *aebcd*

15243 *aebdc*

15324 *aecbd*

15342 *aecdb*

15423 *aedbc*

15432 *aedcb*

21345 *bacde*

21354 *baced*

Приклади на генерування підмножин.

Їх кількість $C_m^n = \frac{m!}{n!(m-n)!}$

Приклад 1.

Знайти всі чотирьохелементні підмножини множини з шести елементів.

Використовуємо формулу:

$$C_6^4 = \frac{6!}{2!\cdot 4!} = \frac{5\cdot 6}{2} = 15$$

1) 1234
2) 1235
3) 1236
4) 1245
5) 1246
6) 1256
7) 1345
8) 1346
9) 1356
10) 1456
11) 2345
12) 2346
13) 2356

14) 2456

15) 3456 → п'ятнадцять підмножин.

Приклад 2.

Знайти всі 3-елементні підмножини 6-елементної множини.

Їх кількість $C_6^3 = \frac{6!}{3!\cdot 3!} = \frac{4\cdot 5\cdot 6}{6} = 20$

1) 123
2) 124
3) 125
4) 126
5) 134
6) 135
7) 136
8) 145
9) 146
10) 156
11) 234
12) 235
13) 236
14) 245
15) 246
16) 256
17) 345
18) 346
19) 356
20) 456

двадцять підмножин.

Література

1. Нікольський Ю. В., Пасічник В.В., Щербина Ю.М. Дискретна математика, Київ, Видавнича група BHV, 2007.
2. Новиков Ф.А.Дискретная математика для программистов, учебник, Санкт-Петербург, Изд. дом «Питер», 2001.
3. Ловас Л., Пламмер М. Прикладні задачі теорії графів. - М.: Світ, 1998.
4. В.Липский "Комбинаторика для программистов", 1982.
5. Н.Кристофидес «Теория графов. Алгоритмический поход».
6. А.А.Зыков "Основы теории графов", 1987.
7. В.А.Евстигнеев "Применение теории графов в программировании", 1985.
8. О.Оре "Графы и их применение", 1965.
9. Э.Рейнгольд, Ю.Нивергельт, Н.Део "Комбинаторные алгоритмы. Теория и практика", 1980.
10. М.Свами, К.Тхуласираман "Графы, сети и алгоритмы", 1984.
11. Зубов В.С. Справочник программиста. Базовые методы решения графовых задач и сортировки. - М.: Информационно-издательский Дом "Филинъ",. 1999.
12. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир, 1978.
13. Липский В. Комбинаторика для программистов. М.: Мир, 1988.

ЗМІСТ

Вступ. Мета і завдання курсу …….. 3

Опис предмета навчальної дисципліни…… 4

Навчально-тематичний план дисципліни…… 5

Зміст лекційних тем курсу…… 6

Теми лабораторних занять…….. 7

Теми самостійної роботи студентів……. 8

Перелік питань на модуль І…… 9

Перелік питань на модуль ІІ…… 10

Перелік питань на залік…..…… 10

Теоретичні відомості:

Основні алгоритми в комбінаториці…… 12

Генерація перестановок…… 43

Генерування перестановок в лексикографічному порядку..44

Генерування перестановок в антилексикографічному порядку..49

Генерування підмножин…… 52

Генерування сполучень…….. 57

Література…… 70

Копча-Горячкіна Галина Ернестівна – викладач кафедри інформаційних управляючих систем та технологій факультету інформатики Закарпатського державного університету

ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ В КОМБІНАТОРИЦІ ТА ГРАФАХ

Навчально-методичний посібник. Частина І. для студентів факультету інформатики напряму „Комп'ютерні науки" зі спеціальності „Програмне забезпечення автоматизованих систем" Закарпатського державного університету

Відповідальний за випуск: **Лавер О.Г.** – кандидат фіз.-мат. наук, завідувач кафедри програмного забезпечення автоматизованих систем та фіз.-мат. Дисциплін.