УДК 373.5.016:004

Горошко Юрій Васильович
доктор педагогічних наук, професор, завідувач кафедри інформатики і обчислювальної техніки
Національний університет «Чернігівський колегіум» імені Т. Г. Шевченка, м. Чернігів, Україна
ORCID ID 0000-0001-9290-7563
horoshko_y@ukr.net

Міца Олександр Володимирович
кандидат технічних наук, доцент, завідувач кафедри інформаційних управляючих систем та технологій
ДВНЗ "Ужгородський національний університет", м. Ужгород, Україна
ORCID ID 0000-0002-6958-0870
alex.mitsa@uzhnu.edu.ua

Мельник Валентин Іванович
заслужений учитель України, викладач інформатики
Кременчуцький педагогічний коледж імені А.С.Макаренка, м. Кременчук, Україна
ORCID ID 0000-0002-2884-2416
melnik7@gmail.com

МЕТОДИЧНІ ПІДХОДИ ДО РОЗВ'ЯЗУВАННЯ ОЛІМПІАДНИХ ЗАДАЧ З ІНФОРМАТИКИ

Анотація. У статті наведено особливості олімпіадних задач з інформатики: відволікаюча фабула, розміщення різних важливих складових завдання в різних місцях умови, нестандартні математичні моделі, нестандартне поєднання стандартних підходів тощо. З огляду на досить високу складність таких задач є потреба в напрацюванні методичних підходів до навчання розв'язувати такі задачі. Розглянуто загальні схеми розв'язування олімпіадних задач з інформатики, запропоновані різними науковцями, що беруть участь в олімпіадному русі. На основі власного досвіду відібрано одну з них. Розглянуто один із напрямів динамічного програмування – так звані *задачі про ранець*. Наведено різні модифікації задачі про ранець, які необхідно вміти розв'язувати для розуміння розв'язку більш складної задачі, що належить до динамічного програмування. Для цих задач наведено відповідні математичні формули або програмний код. Наведено всі етапи застосування вибраної схеми до розв'язування конкретної олімпіадної задачі з інформатики, що належить до класу задач про ранець у двовимірному варіанті й була запропонована одним із авторів на дванадцятій відкритій міжнародній студентській олімпіаді з програмування імені С. О. Лебедєва та В. М. Глушкова "KPI-OPEN 2017": аналіз умови, побудова математичної моделі, побудова загальної схеми розв'язування, уточнення, реалізація, тестування й налагодження, відправлення програми на перевірку. Продемонстровано ефективний авторський метод розв'язування цієї задачі. Наведено програмний код її розв'язку мовою C++. Зазначено, що важливим моментом у підготовці до олімпіад з інформатики є аналіз задач після завершення кожного змагання. Застосування запропонованих методичних підходів до підготовки учнів чи студентів до олімпіад з інформатики (програмування), на нашу думку, дозволить підвищити ефективність такої підготовки.

Ключові слова: олімпіада з інформатики; загальна схема розв'язування задач шляхом написання програми; мова програмування C++.

1. ВСТУП

Постановка проблеми. Розв'язування будь-якої олімпіадної задачі, зокрема з інформатики, є складним і багато в чому творчим процесом. Однак упродовж років різні вчені розробили більш-менш загальні схеми, використання яких допомагає розв'язати задачі з інформатики шляхом створення програми. Один із варіантів такої схеми можна прочитати, наприклад, тут [1]. Але олімпіадні задачі мають суттєві нюанси, і тому ці загальні схеми потребують певного уточнення. Часто трапляються в

умовах олімпіадних задач такі особливості, як відволікаюча фабула; розміщення різних важливих складових завдання в різних місцях умови; нестандартні математичні моделі, які часто потрібно придумати, а не скористатися готовими; нестандартне поєднання стандартних підходів тощо [2], [3]. У реальних змаганнях до особливостей можна віднести нервову напругу, обмежений час на розв'язування, необхідність реалізації стратегії в послідовності розв’язування задач і т.д.

Тому загальна схема потребує уточнення з урахуванням особливостей олімпіадних задач. Науковці вже пропонували уточнення загальної схеми, наприклад, у [4], розв’язування конкретної олімпіадної задачі було також проаналізоване та використане для демонстрації [5]. Але в дослідженні [5] була вибрана досить проста олімпіадна задача, а процес застосування уточненої загальної схеми, на нашу думку, досить схематичний. Інший варіант уточнення загальної схеми щодо розв’язування олімпіадних задач із інформатики наведено в [6]. Тому необхідно навести обидві схеми та обрати, на нашу думку, кращу з них і продемонструвати детальне поетапне її застосування до розв’язування реальної, причому досить складної олімпіадної задачі з інформатики. Для того, щоб учні чи студенти змогли розібратися з цією задачею, потрібна певна пропедевтика, тому спочатку слід розглянути декілька простіших задач із цієї теми. Проведений аналіз показав, що відомі підходи не дозволяють отримати ефективний розв’язок самої задачі, тому був розроблений унікальний авторський підхід.

Кожному тренеру важливо не тільки використовувати загальну схему для підготовки учнів (студентів) до різного виду змагань з програмування, а й мати набір задач, зокрема й складних, для навчання сучасних методів і підходів в олімпіадному програмуванні [7]. Саме таку задачу розглянуто в цій роботі.

Метою статті є розгляд загальних схем розв'язування олімпіадних задач з інформатики, добір та доопрацювання найкращої, на нашу думку, з цих схем; наведення пропедевтики щодо розв'язування однієї зі складних олімпіадних задач з динамічного програмування (так званої задачі про ранець у двовимірному випадку); використання обраної та доопрацьованої схеми для демонстрації авторського підходу до розв'язування цієї задачі.

2. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У роботі [6] наведено одну із загальних схем розв’язування олімпіадних задач з інформатики. Ця схема складається з 7 етапів і наведена у вигляді настанов учневі (студенту).

1. Опишіть рішення кількома реченнями. Мета – це чітко зрозуміти ідею задачі.
2. Знайдіть гарне графічне зображення алгоритму. Це може допомогти розібратися в алгоритмі й привертає увагу до спеціальних випадків або можливих оптимізацій.
3. За допомогою цього графічного представлення спробуйте виконати алгоритм вручну на наборі прикладів. Це дає краще відчуття того, як він працює, і може привести до ідей щодо реалізації алгоритму.
4. Шукайте контрприклади щодо алгоритму, начебто це рішення конкурентів, щодо якого ви хочете довести, що воно неправильне. Якщо контрприклад буде знайдений, то часто він є джерелом нових кращих ідей.
5. Якщо контрприкладів не знайдено, може стати зрозумілим, чому алгоритм є правильним.

6. Визначте складність алгоритму щодо часу та пам'яті. На цьому кроці можна зробити висновок, чи потрібно шукати більш ефективне чи простіше рішення.
7. Напишіть псевдокод алгоритму, спробуйте його спростити й протестувати перед тим, як здійснити фактичну реалізацію обраною мовою програмування.

Іншу схему запропонував С.А. Оршанський [4]. Вона складається з семи етапів. Наведемо цю схему з урахуванням власного досвіду в олімпіадному русі.

1. Читання умови. На цьому етапі потрібно уважно прочитати умову, не пропускаючи жодної фрази. Є типові проблеми:

- у ситуації змагання важко бути уважним. Тому потрібно відвести достатньо часу на уважне читання. Неправильне розуміння умови може призвести до того, що учень буде розв'язувати зовсім іншу задачу, а не ту, яку сформульовано в умові;
- як правило, в умові є певний сюжет. Читання його може відволікати, втомлювати, часто розслабляти, але потрібно бути дуже обережним: у сюжеті можуть бути прямо чи опосередковано важливі дані, що стосуються умови;
- ключова умова може бути схована, наприклад, у форматі вихідних даних. Без цієї умови задача може бути зовсім інакшою, проте теж коректною.

2. Побудова математичної моделі. Спроба формалізувати прочитане часто виявляє значну кількість неузгоджень, що виникли через пропущену при читанні чи неправильно зрозумілу важливу фразу. Тому обов'язково потрібно уважно розглянути наведений приклад вхідних і вихідних даних і зрозуміти, чому вхідні відповідають вихідним. Це покращить розуміння умови, можуть також з'явитися нові ідеї. Побудова математичної моделі може допомогти також придумати розв'язок, що буде працювати на абстрактній віртуальній математичній машині без обмежень пам'яті, часу, діапазону значень змінних і втрат точності в дійсній арифметиці.

3. Побудова загальної схеми розв'язування. Схема полягає в переході від розуміння, що зробити, до розуміння, як це зробити. У кожного учня-олімпіадника є певне поняття про "цеглинки", елементарні структурні одиниці програми, що створюється. Різні "цеглинки" мають різні особливості щодо модифікації. На цьому етапі потрібно побудувати розв'язок із "цеглинок". Основні проблеми, які виникають при побудові загальної схеми розв'язування:

- інтуїтивна комбінація декількох складних блоків, внутрішню структуру яких важко охопити людині, на перший погляд може скласти розв'язок задачі, але при уважному розгляді будуть виникати проблеми. Може з'ясуватися, що, наприклад, у задачі, яка розв'язується динамічним програмуванням, відсутня властивість субоптимальності або що малоймовірний, на перший погляд, "гірший випадок" виникає на кожному достатньо великому наборі вхідних даних;
- придумана схема може виявитися неефективною. Найнадійнішим виходом з цієї ситуації є створення принципово більш ефективного розв'язку всієї задачі або певної підзадачі. Можлива альтернатива – вдосконалення цього розв'язку різними алгоритмічними і програмними оптимізаціями – є ризикованою: можна витратити багато часу, але так і не отримати бажаної ефективності;
- не всі задачі розв'язуються побудовою з "цеглинок". Розв'язок задачі може базуватися на математичних ідеях, які потрібно просто придумати;
- синтез розв'язку з "цеглинок" може допомогти швидко придумати розв'язок, який реалізується довго і з великими зусиллями, тоді як для цієї задачі можна краще трохи більше подумати і знайти щось значно простіше.

4. Уточнення. Потрібно ретельно визначити, з яких частин буде складатися програма, які необхідні структури даних тощо. Часто уточнення дозволяє виявити

помилки, неефективність розв'язку тощо. На етапі уточнення необхідно розгорнути "цеглинки", згадати, як пишуться стандартні алгоритми.

5. Реалізація. На цьому етапі власне пишеться програма. Використовують різні підходи: програмування "згори донизу", "знизу догори", їх комбінація. Перший підхід використовують, коли є загальне бачення програми, – тоді пишеться основна частина програми, а підпрограми не реалізуються, тільки узгоджують їх інтерфейс. Підхід "знизу догори" використовується, коли потрібно щось писати, але не зрозуміло, що саме, а час минає. Тоді починають писати те, що знадобиться в будь-якому випадку, – введення-виведення даних, підпрограму сортування тощо. На практиці частіше використовують комбінацію обох підходів.

6. Тестування і налагодження. Після того, як програма відкомпілювалася, потрібно перевірити її правильність. Проблеми можуть бути найрізноманітнішими: неправильний знак у формулі, недостатній або надлишковий розмір масиву тощо; зрештою, розв'язок може бути принципово неправильним або неефективним. Тому програму треба тестувати. Перше правило тестування – перевірити програму на тесті з прикладу, наведеного в умові задачі. Далі потрібно придумувати власні тести. Потрібно придумати багато "маленьких" тестів. Друге правило – потрібно уважно перевіряти, що ж за програмою виведено в тесті. Крім "маленьких" тестів, необхідно завжди перевірити програму на так званому "максимальному" тесті – повністю випадковому великому тесті з максимальними обмеженнями.

7. Відправлення програми на перевірку. Потрібно не забути про дані для налагодження, включення/виключення оптимізації та перевірок на переповнення арифметики, стеку, виходу за межі масиву тощо. Не потрібно чекати відповіді сервера перевірки, якщо відповідь не надійшла відразу. У цей час можна писати розв'язок другої задачі, думати над третьою тощо.

Надалі ми будемо користуватися схемою, запропонованою С. А. Оршанським.

Звичайно, знання схеми не позбавляє від необхідності досить глибоко розбиратися в значній кількості базових алгоритмів щодо різних тем олімпіадного програмування. Наприклад, однією з таких важливих тем є динамічне програмування, а серед задач, що розв'язуються за допомогою динамічного програмування, чільне місце посідають так звані задачі про ранець.

Щоб застосувати обрану схему до розв'язування конкретної олімпіадної задачі на динамічне програмування, яка була запропонована одним із авторів на дванадцятій відкритій міжнародній студентській олімпіаді з програмування імені С. О Лебедєва і В. М. Глушкова "KPI-OPEN 2017"[8], учні (студенти) вже повинні мати певні базові знання як про суть динамічного програмування, так і пов'язані з розв'язуванням задач різних модифікацій про ранець. Виділимо три задачі з відомого Інтернет-ресурсу e-olymp [9], які дозволять олімпіадникам краще орієнтуватись у цій тематиці.

Першою пропонується класична задача про ранець, яка на цьому ресурсі називається "Ранець Аладіна". Умова її є такою:

У печері є предмети n різних типів, причому кількість предметів кожного типу необмежена. Максимальна вага, яку може витримати ранець, рівна W. Кожен предмет типу i має вагу w_i і вартість v_i $(i = 1, 2, ..., n)$.

Знайдіть максимальну вартість набору предметів, вага якого не перевищує W.

Суть динамічного програмування полягає в тому, що на кожному кроці алгоритму для отримання нових даних використовуються дані, отримані на попередньому кроці. Для цієї задачі схема реалізується такою формулою перерахунку:

$$Dp(k,m) = \max_{0 \le x_k \le \left[\frac{m}{w_k}\right]} \left[v_k x_k + Dp(k-1, m - w_k x_k) \right], \quad (1)$$

де $k = \overline{2,n}$, $m = \overline{0,w}$.

Другою пропонується задача, що має назву “Олімпійський багаж”:

Перебуваючи у Лондоні, турист накупив собі, рідним та друзям багато різноманітних сувенірів та подарунків. Проте коли він купував в аеропорту квиток на дорогу додому, то з великим подивом довідався, що загальна вага його багажу не повинна перевищувати *m* кілограмів.

Знаючи кількість придбаних сувенірів та подарунків – *n* та вагу кожного з них a_i, допоможіть йому укомплектувати свій багаж так, щоб він був якомога ближче до граничного і, звичайно ж, не перевищував обмежень для авіарейсу.

Відповіддю буде максимально допустима вага багажу.

Пропонується розв’язок цієї задачі у вигляді коду програми:

```
#include <iostream>

usingnamespacestd;

int m, n, a[300];
boolres[10001];

intmain()
{
cin>> m >> n;
for (inti = 0; i< n; i++)
cin>> a[i];

res[0] = true;
for (inti = 0; i< n; i++)
for (int j = m; j >= a[i]; j--)
if (res[j – a[i]]) res[j] = true;

for (int j = m; j >= 0; j--)
if (res[j]) { cout<< j; return0;}

}
```

Третьою пропонується задача “Ранець із масами”:

Задано n предметів масою $m_1, m_2, ..., m_n$ та вартістю $c_1, c_2, ..., c_n$ відповідно. Ними наповнюють ранець, який витримує вагу не більше *m*. Визначте набір предметів, який можна винести в ранці і який має найбільшу вартість.

Ця задача відрізняється від першої тим, що предмет або кладуть у ранець, або ні. Це задачу спрощує. Це так звана булева задача про ранець.

Знову наведемо код програми-розв’язку цієї задачі:

```
#include<bits/stdc++.h>
usingnamespacestd;
intDp[101][10001], m[101], c[101], res[101];

intmain()
{
int M, n = 0;
cin>> M;
    n++;
   cin>> m[n];
while (cin.peek() != '\n')
   {
          n++;
          cin>> m[n];
   }
for (int i = 1; i <= n; i++)
cin>> c[i];
```

```
for (int i = 0; i <= M; i++)
    {
if (i >= m[1]) Dp[1][i] = c[1];
    }

for (int  p = 2; p <= n; p++)
for (int  i = 0; i <= M; i++)
if (i>= m[p] &&Dp[p – 1][i – m[p]] + c[p] >Dp[p – 1][i])
Dp[p][i] = Dp[p – 1][i – m[p]] + c[p];
elseDp[p][i] = Dp[p – 1][i];

int k = M, cnt = 0;
for (int i = n; i >= 2; i--)
if (Dp[i][k]!=Dp[i–1][k])
        {
cnt++;
res[cnt] = i;
            k = k – m[i];
        }
if (k >= m[1]) cout<< 1 <<endl;
for (int i = cnt; i > 0; i--)
cout<<res[i] <<endl;
}
```

Відзначимо, що важливо також уміти звести класичну задачу про ранець до булевої задачі про ранець.

Тепер можемо перейти до розгляду задачі, про яку згадувалося вище.

Умова задачі

Редакція газети «Вогні Києва» вирішила дати частину однієї своєї сторінки розміром *A*x*B* під рекламу. Оскільки популярність газети дуже висока, то зразу *N* фірм відгукнулись і запропонували редакції розмістити на цій сторінці свої рекламні оголошення. Відомо, що оголошення *i*-ої фірми має вигляд прямокутної форми та розмір P_i на Q_i, за його опублікування фірма готова заплатити C_i грошових одиниць.

Редакція цієї газети має свій стиль оформлення сторінки. Рекламні оголошення вона буде друкувати так: перше оголошення буде починатись у лівому верхньому кутку, друге – з правої нижньої точки першого, третє – з правої нижньої точки другого і т.д. Усі оголошення, які будуть друкуватися, не повинні виходити за межі сторінки (Рис. 1).

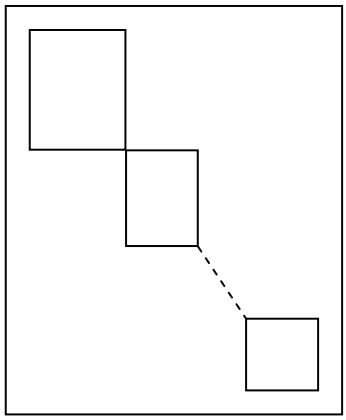

Рис 1. Приклад розміщення рекламних оголошень на сторінці

Фірми запропонували, щоб їхні оголошення друкувалися, якщо на це погодиться редакція, не один раз, а довільну до W_i кількість разів.

Допоможіть редакції визначити, які оголошення і яку кількість разів потрібно друкувати, щоб зароблена сума була максимальною.

Усі дані – цілі числа.

Вхідні дані. У першому рядку знаходяться числа *N, A, B*. Далі в наступних *N* рядках ідуть дані для *i*-го оголошення – C_i, P_i, Q_i та W_i.

Обмеження: 2<=*N*<=20, 2<=P_i<=*A*<=1000, 2<=Q_i<=*B*<=1000, W_i<=100, C_i<=300, час 3 с.

Вихідні дані. Вивести шукану максимальну суму.

Таблиця 1

Приклад

Ввід	Вивід
5 55 55 14 21 10 2 7 17 15 1 12 12 35 1 9 19 23 2 19 39 45 1	40

Застосуємо обрану вище загальну схему до розв'язування цієї задачі.

1. Аналізуючи умову за наведеною схемою, можна помітити, що умова є велика за змістом, і тому важливо з'ясувати в ній ключові моменти. Зокрема, важливо уявити, що мав на увазі автор, і звернути увагу на обмеження, які є в цій задачі, а також з'ясувати, чи застосування їх не дозволить скористатися класичним алгоритмом про ранець у двовимірному випадку, який є для задач такого типу.

2. Побудова математичної моделі.

Учасник олімпіади повинен помітити, що це класична задача на динамічне програмування з двома обмеженнями.

Її математична модель є такою:

$$\sum_{i=1}^{N} P_i x_i \le A, \tag{2}$$

$$\sum_{i=1}^{N} Q_i x_i \le B, \tag{3}$$

$$0 \le x_i \le W_i, \tag{4}$$

$$\text{всі } x_i - \text{цілі}, \ i = 1,\ldots,N; \tag{5}$$

знайти

$$\max_{x_1,\ldots,x_N} f = \sum_{i=1}^{N} C_i x_i \ . \tag{6}$$

3. Побудова загальної схеми розв'язування.

Задамо послідовність функцій

$$Dp(k, m_1, m_2) = \max_{x_1,\ldots,x_k} \sum_{i=1}^{k} C_i x_i \ , \ i = 1,\ldots,N, \tag{7}$$

при обмеженнях

$$\sum_{i=1}^{k} P_i x_i \le m_1 \ , \ \sum_{i=1}^{k} Q_i x_i \le m_2 \ , \ 0 \le x_i \le W_i . \tag{8}$$

Значення m_1 буде змінюватися від *0* до *A*, а значення m_2 буде змінюватися від *0* до *B*. Значення $Dp(k, m_1, m_2)$ показує, яку максимальну суму можна отримати від реклами,

якщо розмір сторінки буде розміром m_1 на m_2 і розглядаються рекламні оголошення перших k фірм.

Уведемо позначення:

$$d_i = \min\left\{\left[\frac{m_1}{P_i}\right],\left[\frac{m_2}{Q_i}\right],W_i\right\},\ i=1,...,N. \tag{9}$$

Знаходити значення першої з функцій (7) можна за формулою:

$$Dp(1,m_1,m_2) = \max_{0\le x_1\le d_1} C_1x_1. \tag{10}$$

Звичайно, при $C_1 \ge 0$ значення $Dp(1,m_1,m_2) = C_1d_1$.

Значення функцій (7) при $k \ge 2$ будемо визначати за формулою:

$$Dp(k,m_1,m_2) = \max_{0\le x_k\le d_k}\left[C_kx_k + Dp(k-1,m_1-P_kx_k,m_2-Q_kx_k)\right]. \tag{11}$$

Максимальною зароблену сумою, згідно з умовою цієї задачі, буде значення:

$$f^* = Dp(N,A,B). \tag{12}$$

Обчислимо оптимальний набір рекламних блоків $x^* = x_1^*, x_2^*,..., x_N^*$ так. Уведемо функцію $X(k,m_1,m_2)$, яка показує кількість повторень k-го блоку при обмеженнях (4). Спочатку обчислимо:

$$X(1,m_1,m_2) = d_1, \tag{13}$$

де d_1 обчислюється за формулою (9).

При $k \ge 2$ обчислення $X(k,m_1,m_2)$ будемо проводити паралельно з обчисленням функції $Z(k,m_1,m_2)$.

Якщо певний варіант $C_kx_k + Z(k-1,m_1-P_kx_k,m_2-V_kx_k)$ виявиться оптимальним, то запишемо:

$$X(k,m_1,m_2) = x_k$$

і перерахуємо:

$$X(i,m_1,m_2) = X(i,m_1-P_kx_k,m_2-V_kx_k),\ i=1,...,k-1. \tag{14}$$

Отже, оптимальний набір рекламних блоків:

$$x_i^* = X(i,A,B),\ i=1,...,N. \tag{15}$$

4. Уточнення.

Наведена вище загальна схема розв'язання має два суттєві недоліки. Перший недолік – це надмірна кількість задіяної для розв'язання задачі оперативної пам'яті, а другий − низька швидкодія програми, реалізованої за цією схемою. Тому ця схема потребує оптимізації, яка полягає ось у чому:

- Функція $Dp(k,m_1,m_2)$ залежить лише від значень функцій $Dp(k-1,x,y)$. Відповідно використовувати перший параметр не обов'язково, досить пам'ятати всі значення попередньої функції $Dp(k-1,x,y)$. Це дозволить зекономити оперативну пам'ять.
- Уведемо умовні додаткові рекламні блоки для кожного з наявних у кількості 1, 2, 4, … (тобто в степенях двійки) до значення, яке б не перевищувало задане значення відповідного W_i. Після чого перейдемо до булевої двовимірної задачі про ранець і будемо робити перерахунок за формулою:

$$Dp(k,m_1,m_2) = \max\left[Dp(k,m_1,m_2),\quad C_k + Dp(k-1,m_1-P_k,m_2-Q_k)\right]. \tag{16}$$

Це суттєво пришвидшує пошук відповіді, незважаючи на збільшення кількості рекламних блоків. Особливо це відчувалося б, якби значення A, B, W_i були значно більшими, ніж є в умові.

5. Реалізація.

Наведемо реалізацію виконання всіх попередніх пунктів загальної схеми мовою С++, яка тепер є загальним стандартом на всіх значних олімпіадах з інформатики.

```
#include <iostream>
#include <vector>
using namespace std;
structRect {                          // створюємо відповідну структуру
intp;   int q;    int c;
Rect(int _p, int _q, int _c) {
        p = _p;        q = _q;        c = _c;
    }
};

vector<Rect>rects;
intdp[1001][1001];

intmain() {
int n, a, b;
cin>> n >> a >> b;
    for (inti = 0; i< n; i++) {
int c, p, q, w;
cin>> c >> p >> q >> w;
int j = 1;
        while (j < w) {
            if (p * j <= a && q * j <= b) {
rects.push_back(Rect(p * j, q * j, c * j)); // заносимо кількості, які є
}                                           // степенями двійки
w -= j;
j *= 2;
}
j = w;
        if (p * j <= a && q * j <= b) {
rects.push_back(Rect(p * j, q * j, c * j));
        }
    }

for (int it = 0; it <rects.size(); ++it) {// робимо перерахунок за
формулою [15]
    for (inti = a; i>= 1; i--) {
        for (int j = b; j >= 1; j--) {
            if (rects[it].p<= i&&rects[it].q <= j) {
dp[i][j] = max(dp[i][j], dp[i - rects[it].p][j - rects[it].q] +
rects[it].c);
            }
        }
    }
}
cout<<dp[a][b] <<endl;;
return 0;
}
```

6. Тестування й налагодження.

Почати тестування потрібно з прикладу, наведеного в умові задачі.

Можна придумати певні “маленькі” тести для цієї задачі.

Таблиця 2

Приклади тестів до задачі

Увід	Вивід
2 89 16 3 8 10 5 3 6 4 5	12
4 401 34 5 3 7 3 5 10 5 3 8 4 2 3 3 1 1 3	53

Щодо великих тестів, то їх краще згенерувати лише для перевірки часу виконання програми, оскільки буде важко перевірити правильність відповіді.

7. Відправлення програми на перевірку.

Усе, що наведено в цьому пункті загальної схеми, для цієї задачі потрібно просто ретельно виконати.

Відзначимо, що важливим моментом у підготовці до олімпіад з інформатики є аналіз задач після завершення кожного змагання, і тренерам потрібно робити відмітки у власному електронному записнику про особливість кожної задачі, з віднесенням її до потрібної категорії. Це, з накопиченням великого масиву розв'язаних задач, робить систему підготовки олімпіадників більш легкою та зручною.

3. ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Отже, у статті було запропоновано як загальні методичні підходи до навчання розв'язування олімпіадних задач з інформатики, так і методику навчання розв'язування класу задач з динамічного програмування, так званих задач про ранець. Наведено ефективний авторський метод розв'язування задачі про ранець у двовимірному випадку. На нашу думку, застосування запропонованих методичних підходів до підготовки учнів чи студентів до олімпіад з інформатики (програмування) дозволить підвищити ефективність такої підготовки, а наведена задача і підхід до її розв'язування стане в пригоді тренерам, що займаються підготовкою учнів (студентів) до різноманітних змагань з програмування.

У подальшому планується застосувати наведені методичні підходи до навчання розв'язування задач з інших розділів олімпіадного програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Ю. Горошко, *Інформаційне моделювання у підготовці майбутніх учителів математики та інформатики.* Чернігів, Україна: Видавець Лозовий В. М., 2012.

[2] В. Мельник, *Задачник з програмування*. Київ, Україна: Видавничий дім "Шкільний світ", 2006.

[3] *Літня школа з програмування (Ужгород, 31 липня – 7 серпня 2016 року) :Матеріали лекцій, умови та розбір задач* . За ред. Олександра Міци, Сергія Оришича. Ужгород: Видавництво "ФОП Сабов А.М.", 2017.

[4] С. Оршанский "О решении олимпиадных задач по программированию формата ACMICPC", *Мир ПК - диск*, №9, 2005.

[5] С. Жуковський, "Аналіз, дослідження та розв'язування конкурсних задач під час учнівської олімпіади з інформатики", *Інформаційні технології в освіті*, №5, с. 152-159, 2010. DOI: 10.14308/ite.

[6] A. Charguéraud, M. Hiron "Teaching Algorithmics for Informatics Olympiads: The French Method" , OLYMPIADS IN INFORMATICS Tasks and Training Volume 2 2008 Selected papers of the

International Conference joint with the XX International Olympiad in Informatics Cairo, Egypt, August 16–23, 2008. [Електронний ресурс]. Доступно: https://www.mii.lt/olympiads_in_informatics/files/volume2.pdf. Дата звернення: Вер. 27, 2018.

[7] В. Мельник, Ю. Горошко, О. Міца "Огляд систем підготовки до олімпіад з інформатики в деяких країнах", *Актуальні питання сучасної інформатики,* №5, с. 21-23, 2017.

[8] Дванадцята відкрита міжнародна студентська олімпіада з програмування імені С. О Лебедєва і В. М. Глушкова "KPI-OPEN 2017" [Електронний ресурс]. Доступно: http://kpi-open.org/. Дата звернення: Лип. 17, 2018.

[9] Інтернет-портал організаційно-методичного забезпечення дистанційних олімпіад з програмування для обдарованої молоді навчальних закладів України e-olimp [Електронний ресурс]. Доступно: https://www.e-olymp.com/. Дата звернення: Лип. 17, 2018.

Матеріал надійшов до редакції 17.07.2018 р.

МЕТОДИЧЕСКИЕ ПОДХОДЫ К РЕШЕНИЮ ОЛИМПИАДНЫХ ЗАДАЧ ПО ИНФОРМАТИКЕ

Горошко Юрий Васильевич
доктор педагогических наук, профессор, заведующий кафедрой информатики и вычислительной техники
Национальный университет «Черниговский коллегиум» имени Т. Г. Шевченко, г. Чернигов, Украина
ORCID ID 0000-0001-9290-7563
horoshko_y@ukr.net

Мица Александр Владимирович
кандидат технических наук, доцент,
заведующий кафедрой информационных управляющих систем и технологий
ГВУЗ "Ужгородский национальный университет", г. Ужгород, Украина
ORCID ID 0000-0002-6958-0870
alex.mitsa@uzhnu.edu.ua

Мельник Валентин Иванович
заслуженный учитель Украины, преподаватель информатики
Кременчугский педагогический колледж имени А.С.Макаренко, г. Кременчуг, Украина
ORCID ID 0000-0002-2884-2416
melnik7@gmail.com

Аннотация. В статье приведены особенности олимпиадных задач по информатике: отвлекающая фабула, расположение различных важных составляющих задания в разных местах условия, нестандартные математические модели, нестандартное сочетание стандартных подходов и т.д. Учитывая это, а также довольно высокую сложность таких задач, возникает проблема наработки методических подходов к обучению решения таких задач. Рассмотрены предложенные различными учеными, принимающими участие в олимпиадном движении, общие схемы решения олимпиадных задач по информатике. На основе собственного опыта выбрана одна из них. Рассмотрено одно из направлений динамического программирования, так называемые задачи о рюкзаке. Приведены различные модификации задач о рюкзаке, умение решать которые необходимо для понимания решения более сложной задачи, относящейся к динамическому программированию. Для этих задач приведены соответствующие математические формулы или программный код. Приведены все этапы применения этой схемы к решению конкретной олимпиадной задачи по информатике, относящейся к классу задач о рюкзаке и предложенной одним из авторов на двенадцатой открытой международной студенческой олимпиаде по программированию имени С. А. Лебедева и В. М. Глушкова "KPI-OPEN 2017": анализ условия, построение математической модели, построение общей схемы решения, уточнение, реализация, тестирование и отладка, отправка программы на проверку. Продемонстрирован эффективный авторский метод решения этой задачи. Приведен программный код решения этой задачи на языке C++. Отмечено, что важным моментом в подготовке к олимпиадам по информатике является анализ задач после завершения каждого соревнования. Применение предложенных методических подходов к подготовке учащихся

или студентов к олимпиадам по информатике (программированию), по нашему мнению, позволит повысить эффективность такой подготовки.

Ключевые слова: олимпиада по информатике; общая схема решения задач путём написания программы; язык программирования C++.

METHODOLOGICAL APPROACHES TO SOLVING OLYMPIAD TASKS ON COMPUTER SCIENCE

Yurii V. Horoshko
Doctor of Pedagogical Sciences, Professor, Head of the Department of Computer Science and Engineering
Taras Shevchenko National University "Chernihiv Collegium", Chernihiv, Ukraine
ORCID ID 0000-0001-9290-7563
horoshko_y@ukr.net

Oleksandr V. Mitsa
PhD of Technical Sciences, Associate Professor,
Head of the Department of Informative and Operating Systems and Technologies
Uzhhorod National University, Uzhhorod, Ukraine
ORCID ID 0000-0002-6958-0870
alex.mitsa@uzhnu.edu.ua

Valentyn I. Melnyk
Honored Teacher of Ukraine, teacher of computer science
Kremenchuk Pedagogical College named after Anton Makarenko, Kremenchuk, Ukraine
ORCID ID 0000-0002-2884-2416
melnik7@gmail.com

Abstract. The article analyzes the peculiarities of the Olympiad tasks on computer science: distracting story, placing various important components of the problem in different places of the condition, non-standard mathematical models, non-standard combination of standard approaches, etc. Taking this into account, as well as the rather high complexity of such tasks, there is the problem of working out methodological approaches to teaching to solve such problems. The general schemes of solving the Olympiad tasks on computer science, proposed by various scientists participating in the Olympiad movement, are considered. Based on the own experience, one of them has been selected. One of the areas of dynamic programming, the so-called Knapsack Problems, is considered. There are given various modifications of Knapsack Problem; the ability to solve them is necessary to understand the solution of a more complex task related to dynamic programming. For these tasks are given appropriate mathematical formulas or program code. There are presented all stages of the application of the given scheme to the solving of a specific Olympiad task on computer science, which belongs to the class of Knapsack Problems and proposed by one of the authors at the Open International Student Programming Olympiad "KPI-OPEN 2017" named after S.O. Lebediev and V.M. Glushkov "KPI-OPEN 2017": the analysis of the condition, the construction of a mathematical model, the construction of a general scheme of solving, refinement, implementation, testing and debugging, sending the program to check. An effective author's method for solving this task is demonstrated. The program code for the solution is given in C++. It is noted that the important point in preparing for the Olympiads on computer science is the analysis of the tasks after the completion of each competition. Applying the proposed methodological approaches to training pupils or students for the Olympiads on computer science (programming), in our opinion, will increase the effectiveness of such training.

Keywords: Olympiad on computer science; general scheme of task solving by writing a program; C ++ programming language.

REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] Y. Horoshko. *Information modeling in the training of future teachers of mathematics and computer science.* Chernihiv, Ukraine: Vydavets Lozovyi V.M., 2012. (in Ukrainian)

[2] V. Melnyk. *Taskbook on programming.* Kyiv, Ukraina: Vydavnychyi dim :Shkilnyi svit", 2006. (in Ukrainian)
[3] *Summer programming school (Uzhhorod, july 31 – august 7, 2016) :Materials of lectures, conditions and analysis of tasks* .Edited by Oleksandr Mitsa, Serhii Oryshych. Uzhhorod: Vydavnytstvo "FOP Sabov A. M.", 2017. (in Ukrainian)
[4] S. Orshanskyi "About the solvingolympiad tasks on programming of format ACMICPC", *Mir P:K - disk*,№9, 2005. (in Russian)
[5] S. Zhukovskyi, "The analysis, research and solution of problems during the students Olympiad in informatics", *Informatsiini tekhnolohii v osviti*, №5, p. 152-159, 2010. DOI: 10.14308/ite. (in Ukrainian)
[6] A. Charguéraud, M. Hiron "Teaching Algorithmics for Informatics Olympiads: The French Method" , OLYMPIADS IN INFORMATICS Tasks and Training Volume 2 2008 Selected papers of the International Conference joint with the XX International Olympiad in Informatics Cairo, Egypt, August 16–23, 2008. [online]. Available: https://www.mii.lt/olympiads_in_informatics/files/volume2.pdf. Accessed on: Sept. 27, 2018. (in English)
[7] V. Melnyk, Y. Horoshko, O. Міца "Overview of the training system for Informatics Olympiads in some countries", *Actual issues of modern informatics,* №5, с. 21-23, 2017. (in Ukrainian)
[8] 12th Open International Student Programming Olympiad "KPI-OPEN 2017" named after S.O. Lebediev and V.M. Glushkov "KPI-OPEN 2017" [online]. Available: http://kpi-open.org/. Accessed on: Jul. 17, 2018. (in Ukrainian)
[9] Internet portal of organizational and methodological support of distance Olympiads on programming for gifted youth of educational institutions of Ukraine e-olimp [online]. Available: https://www.e-olymp.com/. Accessed on: Jul. 17, 2018. (in Ukrainian)