ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ

М. С. Долинский,
Гомельский государственный университет имени Франциска Скорины, Республика Беларусь

ВВЕДЕНИЕ В РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ ЖАДНОГО АЛГОРИТМА

Аннотация

В статье описана методика изучения темы «Жадный алгоритм» при подготовке школьников к олимпиадам по информатике. Технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (http://dl.gsu.by).

Ключевые слова: жадный алгоритм, обучение программированию, олимпиады по информатике, инструментальная система дистанционного обучения.

Контактная информация

Долинский Михаил Семенович, канд. тех. наук, доцент, доцент кафедры математических проблем управления Гомельского государственного университета имени Франциска Скорины, Республика Беларусь; *адрес:* 246000, Республика Беларусь, г. Гомель, ул. Советская, д. 104; *телефон:* (375-232) 77-70-69; *e-mail:* dolinsky@gsu.by

M. S. Dolinsky,
Gomel State University named after Francisk Skorina, the Republic of Belarus

INTRODUCTION TO PROPLEM SOLVING WITH A GREEDY ALGORITHM

Abstract

The article describes the methodology to teach problem solving using a greedy algorithm on Olympiads in informatics. Distance learning system DL.GSU.BY is the effective technical base for teaching.

Keywords: greedy algorithm, teaching for programming, Olympiads in informatics, distance learning tools.

Введение

С сентября 1996 года на базе средней школы № 27 г. Гомеля, Республика Беларусь, а с сентября 1999 года дополнительно и на базе сайта дистанционного обучения DL.GSU.BY ведется работа по факультативному изучению информатики и программирования школьниками разных возрастов [1, 2]. Ключевой особенностью при этом является раннее начало обучения, фактически с первого класса. Как следствие, уже в пятом-шестом классах появляются ученики, которым приходится объяснять такие достаточно сложные темы, как, например, «Жадный алгоритм». Поскольку традиционные подходы рассчитаны на обучение как минимум старшеклассников, то приходится их модифицировать в сторону более простого и наглядного объяснения и более медленного продвижения по учебному материалу, явно выделяя и обозначая все этапы этого продвижения.

В данной статье приводится материал для обучения школьников решению задач по теме «Жадный алгоритм». Как правило, в задачах на эту тему требуется сначала выполнить сортировку, а затем специфический для задачи набор конструктивных действий.

Материал может быть интересен для учителей — как в качестве иллюстрации методики обучения, так и по содержанию, — и в то же время он может оказаться полезным и интересным для школьников и студентов, занимающихся самообучением.

Читателям предлагается следующий порядок работы: откладывать статью в сторону и пытаться самостоятельно выполнить предлагаемое задание: первый раз — после прочтения условия задачи, второй раз — после прочтения указаний к решению.

Задача о заявках

Даны N заявок на проведение занятий в некоторой аудитории. В каждой заявке указаны начало и конец занятия (s_i и f_i соответственно для i-й заявки). В случае пересечения заявок можно удовлетворить лишь одну из них. Заявки с номерами i и j совместны (не пересекаются), если интервалы $[s_i,f_i)$ и $[s_j,f_j)$ не пересекаются (т. е $f_i \le s_j$ или $f_j \le s_i$). Задача состоит в том, чтобы набрать максимальное количество совместных друг с другом заявок.

Формат ввода:
N
$S[1]$ $F[1]$
$S[2]$ $F[2]$
...
$S[N]$ $F[N]$

где:
N — количество заявок;
$S[i]$ $F[i]$ — описание i-й заявки.

Ограничения:
$1 \le N \le 200\,000$;
$0 \le S[i] < F[i] \le 100\,000\,000$;
все числа — целые.

Формат вывода:
Ans — ответ на задачу — максимальное количество совместных друг с другом заявок.

Пример ввода:
5
1 13
6 8
2 4
4 5
7 10

Пример вывода:
3

Это классическая задача «О заявках» в теме «Жадные алгоритмы». Для ее решения достаточно отсортировать заявки в порядке возрастания времени завершения. Включаем первую заявку, а затем последовательно все, которые начинаются не раньше, чем последняя взятая заявка.

В программе это может выглядеть так:

```
Ans:=1; {количество взятых заявок}
j:=1;   {номер последней взятой заявки}
for i:=2 to n do
 if s[i]>=f[j]
  then begin
   inc(Ans);
   j:=i;
  end;
writeln(Ans);
```

Ограничения задачи ($N \le 200\,000$) требуют применения алгоритма быстрой сортировки.

Полный текст решения:

```
const
 MaxN=200000;
var
 s, f: array [1..MaxN] of longint;
 i, N, Ans, j: longint;

procedure QSortF(p, r: longint);
var
 i, j, z, x, t, q: longint;
begin
 if p<r
  then begin
   x:=f[p];
   i:=p-1;
   j:=r+1;
   while i<j do
    begin
     repeat j:=j-1 until f[j]<=x;
     repeat i:=i+1 until f[i]>=x;
     if i<j
      then begin
       t:=f[i]; f[i]:=f[j]; f[j]:=t;
       t:=s[i]; s[i]:=s[j]; s[j]:=t;
      end;
    end;
   q:=j;
   QSortF(p, q);
   QSortF(q+1, r);
  end;
end;
begin
 readln(N);
 for i:=1 to N do readln(s[i], f[i]);
 QSortF(1, N);
 Ans:=1;
 j:=1;
 for i:=2 to n do
  if s[i]>=f[j]
   then begin
    inc(Ans);
    j:=i;
   end;
 writeln(Ans);
end.
```

Задача об отрезках

Имеется N отрезков на прямой ($1 \le N \le 32\,000$). Они начинаются в целочисленных точках и имеют целочисленную длину. Отрезок, покрывающий их все, имеет длину не более 5 300 000 единиц. Определите минимальное количество точек, каждая из которых может быть между своими двумя целочисленными координатами, таких, что на каждом отрезке находится хотя бы одна из этих точек.

Формат ввода (файл *leash.in*):
Строка 1: одно целое число N.
Строки со 2-й по $(N + 1)$-ю: каждая строка содержит два разделенных пробелом положительных числа, описывающих отрезок. Первое число — начало отрезка, второе — его длина.

Пример ввода:
7
2 4
4 7
3 3
5 3
9 4
1 5
7 3

Пояснения.
Графическое изображение данного примера:

```
                  1 1 1 1
1 2 3 4 5 6 7 8 9 0 1 2 3
-------------------------
. 111111111 . . . . . .
. . . 222222222222222 . .
. . 3333333 . . . . . .
. . . . 4444444 . . . . .
. . . . . 555555555
66666666666 . . . . . .
. . . . . 7777777 . . .
```

Формат вывода (файл *leash.out*):
Одно целое число — минимальное количество точек, таких, чтобы каждому отрезку принадлежала хотя бы одна из них.

Пример вывода:
2

Пояснения:

Точка 5,5 принадлежит 1-му, 2-му, 3-му, 4-му и 6-му отрезкам.

Точка 9,5 принадлежит 5-му и 7-му отрезкам.

Возможны и другие варианты с двумя точками. Но невозможно сделать это никакой одной точкой.

Идея решения:

Сортируем отрезки по возрастанию правого края.

Разрезаем по правому краю самого первого отрезка (если правее — его вообще не разрежем).

Помечаем все разрезанные и снова режем по правому краю первого из оставшихся отрезков.

И так до тех пор, пока не разрежем все отрезки.

Полный текст решения:

```
const
 MaxN=32000;
var
 x, L, R: array [1..MaxN] of longint;
 k: array [1..MaxN] of longint;
 N, i, j, ans: longint;

procedure QSort_R_up(pp, rr: longint);
var
 i, j, z, xx, t, q: longint;
begin
 if pp>=rr then exit;
 xx:=R[pp];
 i:=pp-1;
 j:=rr+1;
 while i<j do
  begin
   repeat j:=j-1 until R[j]<=xx;
   repeat i:=i+1 until R[i]>=xx;
   if i<j
    then begin
     t:=R[i]; R[i]:=R[j]; R[j]:=t;
     t:=X[i]; X[i]:=X[j]; X[j]:=t;
    end;
  end;
 QSort_R_up(pp, j);
 QSort_R_up(j+1, rr);
end;

begin
 assign(input, 'leash.in'); reset(input);
 assign(output, 'leash.out'); rewrite(output);
 readln(N);
 for i:=1 to N do readln(x[i], L[i]);
 for i:=1 to N do R[i]:=x[i]+L[i];
 for i:=1 to N do k[i]:=0;
 QSort_R_up(1, N);
 Ans:=0;
 for i:=1 to N do
  if k[i]=0
   then begin
    k[i]:=1;
    inc(ans);
    for j:=1 to N do
     if (R[i]>=x[j]) and (R[i]<=R[j])
      then k[j]:=1;
   end;
 writeln(ans);
 close(input);
 close(output);
end.
```

Далее приводятся решения некоторых задач с целью пополнить арсенал классических жадных алгоритмов использованием специальных функций сравнения.

Задача «Огромное число» (Гомельская городская олимпиада по программированию, ноябрь 2011 года)

Задан набор из N строк, которые состоят из цифр. Необходимо склеить все строки вместе в таком порядке, чтобы полученное число было как можно больше. Найдите это наибольшее число.

Формат ввода:

N
$S[1]$
$S[2]$
...
$S[N]$

где:

N — количество строк;
$S[i]$ — строка, состоящая из цифр.

Ограничения:

$1 \le N \le 1000$;
длина $S[i]$ — менее 100 символов.

Пример ввода:

5
01
11
21
3
33

Формат вывода:

Ответ на задачу — максимальное число, которое получается в результате склеивания исходных строк без ведущих нулей.

Пример вывода:

333211101

Идея решения.

Для того чтобы определить, какую из строк ставить раньше, используем следующую функцию сравнения:

$$s1 < s2, \text{ если } s1 + s2 < s2 + s1.$$

Для решения задачи сортируем строки по убыванию (используя вышеопределенную функцию сравнения строк), а затем выводим строки в получившемся порядке. Перед выводом пропускаем все незначащие нули.

Полный текст решения:

```
const
 MaxN=1000;
var
 s: array [1..MaxN] of string[100];
 i, j, N: longint;
 p: ansistring;

procedure Sort1;
var
 i, j: longint;
 t: string;
begin
 for i:=1 to N do
  for j:=1 to N-1 do
   if (s[j]+s[j+1]) < (s[j+1]+s[j])
    then begin
     t:=s[j];
     s[j]:=s[j+1];
```

```
    s[j+1]:=t;
   end;
end;

begin
 readln(N);
 for i:=1 to N do readln(s[i]);
 Sort1;
 p:='';
 for i:=1 to N do p:=p+s[i];
 i:=1;
 while p[i]='0' do inc(i);
 for j:=i to length(p) do write(p[j]);
 writeln;
end.
```

Задача «Белоснежка и N гномов» (Санкт-Петербургская командная олимпиада по программированию, 2006 год)

У Белоснежки N гномов, и все они очень разные. Она знает, что для того, чтобы уложить спать i-го гнома, нужно a_i минут, и после этого он будет спать ровно b_i минут. Помогите Белоснежке узнать, может ли она получить хотя бы минутку отдыха, когда все гномы будут спать, и если да, то в каком порядке для этого нужно укладывать гномов спать.

Например, пусть гномов всего два и $a_1 = 1$, $b_1 = 10$, $a_2 = 10$, $b_2 = 20$. Если Белоснежка сначала начнет укладывать первого гнома, то потом ей потребуется целых 10 минут, чтобы уложить второго, а за это время проснется первый. Если же она начнет со второго гнома, то затем она успеет уложить первого и получит целых 10 минут отдыха.

Формат ввода:

Первая строка входного файла содержит число N ($1 \le N \le 10^5$), вторая — числа $a_1, a_2, ..., a_n$, третья — числа $b_1, b_2, ..., b_n$ ($1 \le a_i, b_i \le 10^9$).

Формат вывода:

Выведите в выходной файл N чисел — порядок, в котором нужно укладывать гномов спать. Если Белоснежке отдохнуть не удастся, выведите число -1.

Пример ввода:	*Пример ввода:*
2 1 10 10 20	2 10 10 10 10
Пример вывода:	*Пример вывода:*
2 1	-1

Идея решения.

Сортируем гномов по убыванию суммы $a[i] + b[i]$ ($= c[i]$).

Синхронно обмениваем все массивы вместе: c, a, b, Num (здесь Num — это массив первоначальных номеров гномов).

Далее для всех гномов от 1-го до $(N-1)$-го проверяем выполнение свойства: $b[i] > a[i+1]$. То есть время, которое будет спать i-й гном, строго больше времени, которое требуется для того, чтобы уложить $(i+1)$-го гнома.

Если свойство выполнено для всех гномов, выводим нужный порядок (массив Num). Иначе выводим -1.

Решение с обычной сортировкой не проходит по времени, нужна быстрая сортировка:

```
const
 MaxN=100000;
var
 a, b, c, Num: array [1..MaxN] of longint;
 i, N: longint;

procedure QSortC_Down(L, R: longint);
var
 i, j: longint;
 x, t: longint;
begin
 if L>=R then exit;
 x:=C[L];
 i:=L-1;
 j:=R+1;
 while i<j do
  begin
   repeat i:=i+1 until C[i]<=x;
   repeat j:=j-1 until C[j]>=x;
   if i<j
    then begin
     t:=c[i]; c[i]:=c[j]; c[j]:=t;
     t:=a[i]; a[i]:=a[j]; a[j]:=t;
     t:=b[i]; b[i]:=b[j]; b[j]:=t;
     t:=num[i]; num[i]:=num[j]; num[j]:=t;
    end;
  end;
 QSortC_Down(L, j);
 QSortC_Down(j+1, R);
end;

begin
 assign(input, 'dwarfs.in'); reset(input);
 assign(output, 'dwarfs.out'); rewrite(output);
 readln(N);
 for i:=1 to N do read(a[i]);
 for i:=1 to N do read(b[i]);
 for i:=1 to N do c[i]:=a[i]+b[i];
 for i:=1 to N do Num[i]:=i;
 QSortC_Down(1, N);
 i:=1;
 while (i<=N-1) and (b[i]>a[i+1]) do inc(i);
 if i>N-1
  then for i:=1 to N do write(Num[i], ' ')
  else writeln(-1);
 close(input);
 close(output);
end.
```

Список использованных источников

1. *Долинский М. С.* Гомельская школа олимпиадного программирования // Информатика и образование. 2015. № 7.

2. *Долинский М. С., Кугейко М. А.* Гомельская инструментальная система дистанционного обучения // Информатика и образование. 2010. № 11.