n

Бинарна претрага по простору решења

https://arena.petlja.org/sr-Latn-RS/competition/2021binarnaporesenju

1. Дрва

Дрвосеча треба да насече одређену количину дрвета и има тестеру коју може да подешава да сече на било којој целобројној висини (у метрима). Пошто тестера сече само дрво изнад висине на коју је постављена, што је тестера више постављена, насећи ће се мање дрвета. Пошто дрвосеча брине о околини, он не жели да насече више дрвета него што му је потребно. Напиши програм који одређује највишу могућу целобројну висину тестере, тако да дрвосеча добије довољно дрвета (претпостави да увек постоји довољно дрвета).

Улаз

Са стандардног улаза се учитава број дрвећа у шуми n ($1 \leq n \leq 10^5$), а затим низ висина сваког дрвета (низ природних бројева између 1 и 10000, сваки у посебном реду). Након тога учитава се и количина насеченог дрвета (пошто су сва дебла исте дебљине, количина се мери у метрима висине исечених стабала).

Излаз

На стандардни излаз исписати тражену максималну целобројну висину тестере.

Пример

Улаз

```
5
24
21
19
14
22
14
```

Излаз

```
18
```

Објашњење:
постављањем тестере на 18 метара,
од првог дрвета ћемо одсећи 24-18=6 метара, од другог 21-18=3, од трећег 19-18=1,
од четвртог ништа, а од петог 22-18=4 метра.
То је укупно 6+3+1+4=14 метара, што је тачно онолико колико је потребно.

Решење

Прикажимо идеју алгоритма сортирања са инкременталним увећањем на примеру из поставке задатка. Потребно је да насечемо 14 метара дрвета, а висине су [24, 21, 19, 14, 22].
Након сортирања овог низа опадајуће, добијамо низ висина [24, 22, 21,19, 14, 0].

- Ако је висина тестере 24, насећи ћемо 0 метара.
- Ако је висина тестере 22, количина ће се са 0 повећати за 1* (24 - 22) тј. за 2 и биће једнака 2.

- Ако је висина тестере 21, количина ће се повећати за 2 *(22 - 21) тј. за 2 и биће једнака 4.
- Ако је висина тестере 19, количина ће се повећати за 3*(21 - 19) тј. за 6 и биће једнака 10.
- Ако је висина тестере 14, количина ће се повећати за 4*(19 -14) тј. за 20 и биће једнака 30.

Ово је више него што нам је потребно, па ћемо додатно мало подигнути тестеру.
Вишак у односу на потребну количину је 30 - 14 = 16.
Ако то поделимо на 4 дрвета која се секу добијамо

$$\left[\frac{30-14}{4}\right]=4$$

и висину подешавамо на 14 + 4 = 18.

Потребно је само пазити и посебно разматрати случај када се ни постављањем тестере на висину најнижег дрвета не добије довољно дрвета (а то је могуће да се деси). Тада је тестеру потребно спустити до земље, а онда поправку израчунати на горе описани начин. Најлакши начин да се ово имплементира је да се у низ вештачки дода посебно дрво висине 0.

Остаје још питање како одредити количину дрвета која се добија када се тестера постави на висину дрвета на позицији k.
Један начин је да се та количина сваки пут изнова рачуна, но то би било веома неефикасно. Боље решење је да се низ сортира и да се затим количина рачуна инкрементално.
Спуштањем секире на висину дрвета на позицији k од сваког од првих k дрвета одсечено је парче од $h_k - h_{k-1}$ метара, па ако знамо количину дрвета која се исече када је тестера на висини дрвета на позицији k-1, количину дрвета када је тестера на висини дрвета на позицији k можемо веома једноставно добити увећавањем те количине за k* ($h_k - h_{k-1}$).

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

long long naseceno(const vector<int>& visine, int testera) {
  long long rezultat = 0;
  for (int v : visine)
    if (v >= testera)
      rezultat += v - testera;
  return rezultat;
}

int main() {
  int n;
  cin >> n;
  vector<int> visine(n);
  for (int i = 0; i < n; i++)
    cin >> visine[i];
  long long potrebno;
  cin >> potrebno;
```

```
  int testera = 0;
  while (naseceno(visine, testera) >= potrebno)
    testera++;

  cout << testera - 1 << endl;

  return 0;
}
```

Сортирање дрвећа по висини је сложености O(n log n). Након тога се дрво чија је висина довољна да би се добила довољна количина насеченог дрвета одређује једним проласком кроз низ и инкреметанталним ажурирањем количине насеченог дрвета (што је сложености O(1)), па се тражено дрво налази у сложености O(n). Завршна поправка висине врши се у времену O(1). Дакле, сложеношћу доминира сортирање и укупна сложеност износи O(n log{n}).

Друго решење проблема се може засновати на бинарној претрази по решењу
тј. по тражењу оптималне вредности коришћењем бинарне претраге. Постављањем тестере на висину h, код свих дрва која су виша од h биће одсечено h_i - h метара, док од осталих дрва неће бити исечено ништа. На основу тога, за фиксирану висину тестере грубом силом (испитивањем сваког дрвета засебно) у времену O(n) можемо израчунати укупну количину насеченог дрвета. Бинарна претрага је применљива јер знамо да је до одређених висина тестере дрвета довољно, а да је од одређене висине тестере дрвета премало, тако да заправо тражимо преломну тачку, тј. највећу висину тестере за коју је дрвета довољно тј. последњи елемент низа који задовољава услов.

Ако је максимална висина дрвета M, тада је сложеност овог приступа $O(n log{M}). Наиме, бинарном претрагом се претражује интервал [0, M], па се провера да ли је насечено довољно дрвета позива log{M} пута.
Израчунавање количине насеченог дрвета и провера да ли је она довољна врши се једним пролазак кроз низ дрвета и сложености је O(n).

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int testera(const vector<int>& visine, int potrebno) {
  int od_visina = 0;
  int do_visina = *max_element(begin(visine), end(visine));
  while (od_visina <= do_visina) {
    int visina = od_visina + (do_visina - od_visina) / 2;
    long long naseceno = 0;
```

```
    for (int v : visine)
      if (v >= visina)
        naseceno += v - visina;

    if (naseceno >= potrebno)
      od_visina = visina + 1;
    else
      do_visina = visina - 1;
  }
  return do_visina;
}

int main() {
  int n;
  cin >> n;
  vector<int> visina(n);
  for (int i = 0; i < n; i++)
    cin >> visina[i];
  long long potrebno;
  cin >> potrebno;

  cout << testera(visina, potrebno) << endl;

  return 0;
}
```

2. Конференција

Током једног дана конференције одржава се n предавања. Представља се m рачунарских компанија које су означене бројевима од 1 до m, при чему неке компаније држе и више предавања. Један од учесника конференције жели да присуствује предавањима свих компанија без напуштања сале за предавања. Написати програм који одређује који је најмањи број предавања које мора да одслуша да би чуо све компаније.

Улаз

На стандардном улазу у првом реду је број компанија m ($1 \leq m \leq 50$), у другом реду је укупан број предавања n ($1 \leq n \leq 50000$), а у трећем је наведен редослед предавања компанија тако што су наведени њихови редни бројеви (од 1 до m) раздвојени размацима.

Излаз

Најмањи број предавања које учесник треба да одслуша у континуитету да би чуо бар по једно предавање сваке компаније.

Ако није могуће да учесник одслуша предавање сваке компаније, исписати текст “ne moze” (без наводника).

Пример 1

Улаз

```
4

20

4 2 4 3 3 2 2 4 2 2 3 3 1 3 3 1 4 4 1 4
```

Излаз

```
6
```

Објашњење

Могуће је одслушати предавања компанија `4 2 2 3 3 1`

Пример 2

Улаз

```
3

10

1 2 1 2 1 2 1 2 1 2
```

Излаз

```
ne moze
```

3. Најкраћа подниска која садржи све дате карактере

Графички дизајнер је преуредио неколико слова у једном фонту и жели да своје промене прикаже клијенту. У дугачком тексту је потребно да одабере најкраћи део (сегмент узастопних слова) који садржи сва слова која је променио.

Улаз

У првој линији стандардног улаза налази се текст (једноставности ради претпоставимо да је састављен само од малих слова енглеског алфабета) чија је дужина највише 50000 карактера. У другој линији се налази скуп слова (опет, претпоставимо малих слова енглеског алфабета) које је дизајнер променио (слова су написана једно до другог, без размака и без понављања).

Излаз

На стандардни излаз исписати један цео број који представља дужину најкраћег дела текста који садржи све карактере датог скупа. Ако такав део текста не постоји, исписати `nema`.

Пример 1

Улаз

```
dobardansvimakakoste

arnk
```

Излаз

```
10
```

Пример 2

Улаз

```
abababababab

abc
```

Излаз

```
nema
```

4.

Највећи квадрат у хистограму

Напиши програм који одређује површину највећег квадрата који се може уписати у задати хистограм (хистограм се састоји од стубаца ширине 1).

Улаз

Са стандардног улаза се уноси број стубаца n ($1 \leq n \leq 50000$), а затим и висине стубаца (позитивни цели бројеви мањи од 10_5, раздвојени размацима).

Излаз

На стандардни излаз исписати тражену површину.

Пример

Улаз

```
5

1 5 4 4 2
```

Излаз

```
9
```

Објашњење: Највећи квадрат је означен карактерима +.

```
  *

  ***

  +++

  +++*

*+++*
```

5.

Гласници

Дуж једног пута налазе се гласници. Поруку први сазнаје гласник на почетку пута и циљ је да сви гласници што пре сазнају поруку. Сваки гласник може викањем пренети поруку свима који се од њега налазе на растојању мањем од

задатог домета гласа (истог за све гласнике). При том се сви гласници могу кретати брзином од највише једног метра у секунди у било ком смеру (током времена могу мењати своју брзину и смер кретања, а могу и стајати у месту).

Улаз

Са стандардног улаза се учитава домет гласа d (реалан број, $1 \leq d \leq 10^6$), затим број гласника n (цео број, $1 \leq n \leq 10^5$), а након тога положај сваког гласника g_i (реалан број, $0 \leq g_i \leq 10^9$, који представља растојање од почетка пута).

Излаз

На стандардни излаз исписати најмање време протекло од тренутка када први гласник сазна поруку до тренутка када поруку сазнају сви гласници, као реалан број заокружен на три децимале (одступање од тачне вредности сме да буде највише 10^{-3}).

Пример 1

Улаз

```
3.000
2
0.000
6.000
```

Излаз

```
1.500
```

Пример 2

Улаз

```
2.000
4
0.000
4.000
4.000
8.000
```

Излаз

```
1.000
```