Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3231

Савон Г.К.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2021

Задача E «Коровы в стойла»

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

using namespace std;

vector<int> stoila_koord;
int k;
int n;
long long int sum;

vector<int> try_put_cows(long long int dist) {
  int now_dist = 0;
  int count_cows = 1;
  int now_stoilo = -1;
  long long int min_dist = sum;
  vector<int> ans;
  ans.clear();
  while (count_cows < k && now_stoilo < n - 2) {
    now_stoilo++;
    now_dist += stoila_koord[now_stoilo];
    if (now_dist >= dist) {
      count_cows++;
      if (now_dist < min_dist) min_dist = now_dist;
      now_dist = 0;
    }
  }
  ans.push_back(count_cows);
  ans.push_back(now_stoilo);
  ans.push_back(min_dist);
  return ans;
}

void bin(long long int l, long long int r) {
  long long int mid;
  vector<int> result_of_try;  // 0 - count_cows, 1 - count_sloila, 2-min dist
  while ((r - l) > 1) {
    result_of_try.clear();
    mid = ((r + l) / 2);
    result_of_try = try_put_cows(mid);
    if (result_of_try[0] == k && result_of_try[1] == n - 2) {
      if (result_of_try[2] == mid) {
        cout << mid;
        return;
      } else {
        l = mid;
      }
    } else {
      if (result_of_try[0] < k && result_of_try[1] == n - 2) r = mid;
      if (result_of_try[0] == k && result_of_try[1] < n - 2) l = mid;
    }
  }
  cout << l;
}
```

```
int main() {
  cin >> n;
  cin >> k;
  int koord_prev;
  int koord;
  sum = 0;
  cin >> koord;
  for (int i = 1; i < n; i++) {
    koord_prev = koord;
    cin >> koord;
    stoila_koord.push_back(koord - koord_prev);
    sum += koord;
  };
  bin(1, sum);
  return 0;
}
```

Пояснение к примененному алгоритму:

Честно сказать, когда я прочитала эту задачу, то поняла, что когда-то решала похожее в школе, поэтому сразу стала решать бинпоиском по ответу. Поэтому вряд ли расскажу душетрепещащую историю о пути создания решения.

Но вообще в целом вес логично, мы понимаем, что расстояние между коровами не может быть больше расстояния между первым и последним столбом давайте сразу назовем это расстояние SUM (на самом деле даже меньше в зависимости от количества коров, но все же). А также это расстояние не может быть меньше 1 (ну или если еще чуть оптимизировать, то меньше, чем минимальное расстояние между столбами, хотя можно, наверное, еще более точно сказать, но ладно).

Дальше пытаемся распихать всех этих коров со средним расстоянием, и смотрим, что же раньше закончится – коровы или стойла. В зависимости от этого понимаем – вынуждены ли мы уменьшать предполагаемый ответ или увеличивать.

Ну и изменяем соответственно левый и правый край по всем законам бинпоиска.

Сложность: O(log(sum))

Задача F «Число»

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

using namespace std;
```

```
vector<vector<string> > inputs;

bool comp(string s1, string s2) {
  string new_str1 = s1 + s2;
  string new_str2 = s2 + s1;
  if (new_str2 > new_str1) return 1;
  return 0;
}

int main() {
  string input;
  ifstream cin;
  cin.open("number.in");
  inputs.resize(10);
  while (cin >> input) {
    inputs[input[0] - '0'].push_back(input);
  }
  cin.close();
  for (int i = 0; i <= 9; i++) {
    sort(inputs[i].begin(), inputs[i].end(), comp);
  }
  ofstream cout;
  cout.open("number.out");
  for (int i = 9; i >= 0; i--)
    for (int j = inputs[i].size() - 1; j >= 0; j--) cout << inputs[i][j];
  cout.close();
  return 0;
}
```

Пояснение к примененному алгоритму:

Сразу ясно, что чем больше первая цифра числа – тем раньше надо это число поставить, поэтому сразу при считывании я пихаю в разные векторы числа с разной первой цифрой, чтоб потом лишний раз не сравнивать все числа, начинающиеся с 9 со всеми, начинающимися с 0 например.

А после нам надо отсортировать все элементы каждого из векторов, так, чтобы итоговое число получалось как можно больше. Сначала я начала писать супер-сложный компаратор, которые поочередно сравнивает каждую циферку, пока не найдет разницу, а потом, если вдруг одно из чисел вдруг неожиданно закончилось, то тоже надо проверять еще всякого, в общем намудрила жестко, долго думала, что с этим делать.

Но потом поняла, что можно куда проще делать. Просто переставляем эти числа и сравниваем два полученных больших числа, и соответственно в таком порядке при сортировке и оставляем. И тогда париться о каждой конкретной цифре не нужно вообще. Вот так вот.

Сложность: O(nlog(n)), где n количество бумажек (причем можно даже сказать, что O(mlog(m)), где m – максимальное количество бумажем в каком-нибудь из векторов)

Задача G «Кошмар в замке»

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

using namespace std;

vector<vector<int> > letters;  // 0 - сама буква, 1 - ее вес, 2 - ее количество;

bool comp(vector<int> vec1, vector<int> vec2) {
  if (vec2[1] >= vec1[1]) return 0;
  return 1;
}

int main() {
  string stroka;
  cin >> stroka;
  vector<int> little_vec;
  int ves;
  for (int i = 0; i < 26; i++) {
    little_vec.clear();
    little_vec.push_back(i);
    cin >> ves;
    little_vec.push_back(ves);
    little_vec.push_back(0);
    letters.push_back(little_vec);
  }
  for (int i = 0; i < stroka.size(); i++) {
    letters[stroka[i] - 'a'][2]++;
  }
  sort(letters.begin(), letters.end(), comp);
  for (int i = 0; i < letters.size(); i++) {
    if (letters[i][2] > 1) {
      cout << (char)(letters[i][0] + (int)('a'));
    }
  }
  for (int i = 0; i < letters.size(); i++) {
    if (letters[i][2] == 1) {
      cout << (char)(letters[i][0] + (int)('a'));
      letters[i][2]--;
    } else {
      for (int j = letters[i][2]; j >= 3; j--)
        cout << (char)(letters[i][0] + (int)('a'));
    }
  }
  for (int i = letters.size() - 1; i >= 0; i--) {
    if (letters[i][2] > 0) cout << (char)(letters[i][0] + (int)('a'));
  }
  return 0;
}
```

Пояснение к примененному алгоритму:

В замке действительно разброд и шатания. И я сначала подумала, что условия чуть сложнее. И, например, в строке adfhaoa – максимальное расстояние для a – не между первой и третьей, а между первой и второй и это куда более сложная задача. А в итоге все гораздо прозаичнее.

Все буквы у которых нет пары нам вообще не важны, поэтому их мы запихнем в середину, для увеличения расстояния между парными, все оставшиеся от пары(в смысле для буквы, количество которой больше 1, количество минус 2 для пары) – туда же.

А остальные, в соответствии с весом расставляем по краям. Чем больше вес, тем это пара занимает более «наружнее» место, чтобы расстояние было как можно больше.

Сложность: (nlog(n))

Задача H «Магазин»

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

using namespace std;

int main() {
  int n;
  int k;
  cin >> n >> k;
  vector<int> prises;
  int prise;
  for (int i = 0; i < n; i++) {
    cin >> prise;
    prises.push_back(prise);
  }
  sort(prises.begin(), prises.end());
  long long int ans = 0;
  int iter = 1;
  for (int i = prises.size() - 1; i >= 0; i--) {
    if (iter != k) {
      ans += prises[i];
      iter++;
    } else
      iter = 1;
  }
  cout << ans;
  return 0;
}
```

Пояснение к примененному алгоритму:

Ясно, что бесплатно пробьют определенное число товаров. И нам надо сделать так чтобы эти товары в сумме стоили как можно больше.

Если скидка распространяется, на k-ый товар, то ясно, что самые дорогие k-1 товар получить без скидки у нас не выйдет никак, а вот k-ый по дороговизне – очень даже – если пихнем в чек вместе с первыми k-1 дорогущими, за которые уж мы полюбому заплатим – и это конечно самое выгодное, что мы – предмет ненависти всей очереди в магазине – можем сделать.

После того как пробили первый чек и порадовались тому, что получили k-ый бесплатно – отбрасываем первые k товаров и про оставшиеся проводим те же рассуждения.

Сложность: O(nlog(n))