Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Алгоритмы и структуры данных

Отчет по первому блоку задач (Timus)

Выполнил:

Марьин Григорий Алексеевич

Группа P3212

Проверил:

Голиков Андрей Сергеевич

Санкт-Петербург

2026

1115. Корабли

Суть задачи:

Нужно разбить числа на m различных сумм.

Решение:

Будем рекурсивно перебирать варианты. Для начала отсортируем корабли в обратном порядке, чтобы сначала “впихнуть” большие. В рекурсию будем передавать номер ряда, для которого мы хотим составить сумму, набранную уже сумму, индекс, с которого мы смотрим корабли. При переходе к следующей итерации смотрим, может ли корабль с i-тым индексом влезть в ряд, если да, и при этом еще осталось место, то заходим в следующий этап рекурсии, но нужно набрать уже сумму не s, а s+v[i], где v[i] - длина корабля, который только что взяли, также отметим этот корабль как использованный. Когда s станет равной нужной для ряда, то перейдем к следующему ряду. Если не удалось набрать сумму нужную, то уберем последний добавленный корабль и так далее. Алгоритм работает, тк происходит полный перебор всех вариантов разбиений, а значит один да найдется.

Алгоритмическая сложность:

- **Время:** худший случай $O(2^n)$
- **Память:** O(n)

Код:

```
#include <bits/stdc++.h>

using namespace std;

int n, m;
vector<int> v;
vector<int> rows;
vector<int> used;
vector<vector<int>> ans;

bool solve(int row, int cur_sum, int st) {
 if (row == m) {
   return true;
 }

 if (cur_sum == rows[row]) {
   return solve(row + 1, 0, 0);
 }

 for (int i = st; i < n; i++) {
   if (!used[i]) {
     if (cur_sum + v[i] <= rows[row]) {
       used[i] = 1;
       ans[row].push_back(v[i]);
       if (solve(row, cur_sum + v[i], i + 1)) {
         return true;
       }
       ans[row].pop_back();
       used[i] = 0;
     }
   }
```

```
  }
  return false;
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(nullptr);

  cin >> n >> m;

  v.resize(n);
  rows.resize(m);
  used.resize(n);
  ans.resize(m);

  for (int i = 0; i < n; ++i) {
    cin >> v[i];
  }

  for (int i = 0; i < m; ++i) {
    cin >> rows[i];
  }

  sort(v.rbegin(), v.rend());

  if (solve(0, 0, 0)) {
    for (int i = 0; i < m; ++i) {
      cout << ans[i].size() << '\n';
      for (size_t j = 0; j < ans[i].size(); ++j) {
        cout << ans[i][j] << ' ';
      }
      cout << '\n';
    }
  }
  return 0;
}
```

1296. Гиперпереход

Суть задачи:

Найти отрезок с максимальной суммой.

Решение:

Будем решать динамическим программированием. dp[i] - максимальная сумма, которую можно накопить к i-тому индексу. Переход: dp[i] = max(dp[i-1]+v[i], 0). Если новое число уменьшить сумму, что она станет меньше нуля, то его точно не стоит брать. Ответ будет - максимальное число в массиве dp.

Алгоритмическая сложность:

- **Время:** O(n)
- **Память:** O(n)

Код:

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long int ll;
int main() {
 int n;
 cin >> n;
 if (n == 0) {
   cout << 0;
   return 0;
 }
 vector<int> v(n);
 for (int i = 0; i < n; i++) {
   cin >> v[i];
 }
 vector<int> dp(n);
 dp[0] = max(v[0], 0);
 for (int i = 1; i < n; i++) {
   dp[i] = max(dp[i - 1] + v[i], 0);
 }
 int ans = 0;
 for (int i = 0; i < n; i++) {
   ans = max(ans, dp[i]);
 }
 cout << ans;
 return 0;
}
```