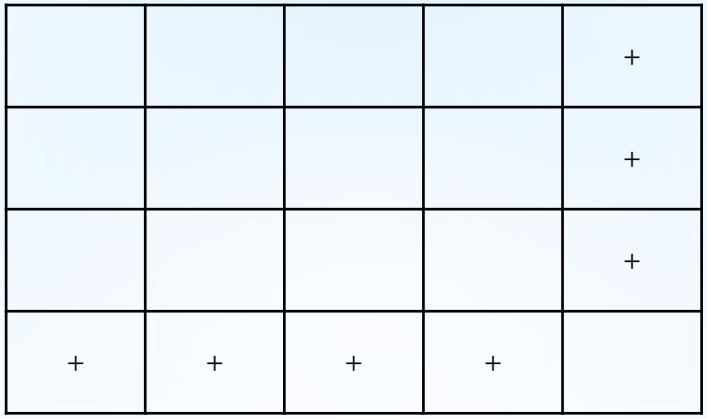
Вологодская область
II Областная олимпиада школьников по информатике
2017-2018 учебный год
9-10 классы
Заключительный тур

Разбор задач

Задача 1 – Цветочная клумба

Ответ: M + N - 2

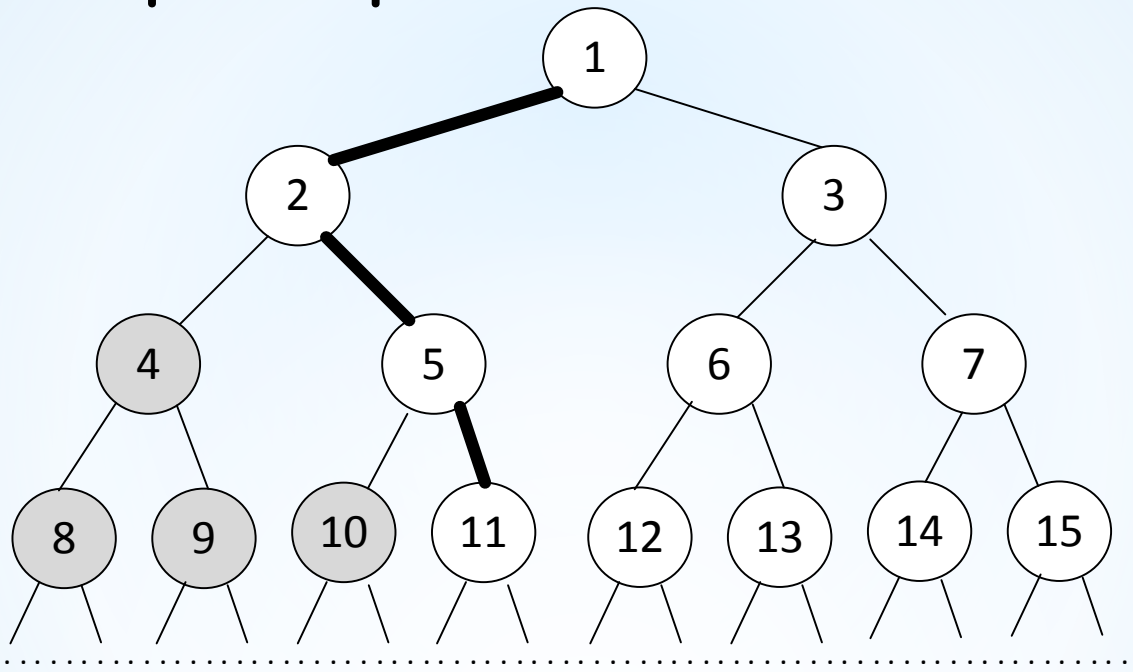


Докажем, что не существует ответа больше, чем M + N - 2

Для каждого посаженного цветка либо в его строке, либо в его столбце нет других цветков (так как если другие цветки будут и в строке, и в столбце, то можно построить прямоугольный треугольник).

Максимум мы можем сделать (M-1) строку с одним цветком и (N-1) столбец с одним цветком, как показано на рисунке.

Задача 2 – Бинарное дерево



1). Заметим, что номера вершин на пути от N до корня – это N div 1 , N div 2, N div 4, N div 8 и т.д. :

11 div 1 = 11, 11 div 2 = 5, 11 div 4 = 2, 11 div 8 = 1

2). Заметим, что номера самых левых вершин – это степени двойки

Осталось лишь просуммировать их разности.

При этом, если при движении к корню мы пришли в степень числа 2, то ответ уже найден. Например, для N=1048579 ответ = 1048579-1048576+524289-524288 = 4

Ответы на вопросы задачи: 15 70 0 16369 4

Задача 3 – Ребус

ANT+ACT=CAT

180+130=310
270+250=520
360+370=730
450+490=940

Идея решения - перебор вариантов:

Перебираем первое и второе число, находим их сумму (она не должна превышать 999).

Каждое из трёх чисел разбиваем на цифры.

Проверяем, что:

а). одинаковым буквам соответствуют одинаковые цифры

б). разным буквам соответствуют разные цифры

Решение задачи «Ребус» на языке C++:

```
#include <stdio.h>

char s[12], w[10], d[9];

void fill(int a, char d[]) {
  d[0] = a / 100; d[1] = a / 10 % 10; d[2] = a % 10;
}

bool chk() {
  for (int i = 0; i < 9; i++)
    for (int j = 0; j < 9; j++)
      if (w[i] == w[j] && d[i] != d[j] || w[i] != w[j] && d[i] == d[j])
          return false;
  return true;
}

int main() {
  scanf("%c%c%c+%c%c%c=%c%c%c",w,w+1,w+2,w+3,w+4,w+5,w+6,w+7,w+8);
  for (int a = 0; a <= 999; a++) {
    fill(a, d);
    for (int b = 0; b <= 999 - a; b++) {
      fill(b, d + 3);
      fill(a + b, d + 6);
      if (chk())
        printf("%d%d%d+%d%d%d=%d%d%d\n",d[0],d[1],d[2],d[3],d[4],d[5],d[6],d[7],d[8]);
    }
  }
}
```

Задача 4 - «Выгодный бизнес» - частичные решения

Подзадача 1: перебор всех интервалов с явным вычислением суммы на каждом.
Сложность $O(N^3)$

Подзадача 2: вычисление префиксных сумм.
Перебор всех интервалов, вычисление суммы на каждом за O(1).
Сложность $O(N^2)$.

```
#include <stdio.h>
#include <vector>
#include <algorithm>

int main() {
  int n;
  scanf("%d", &n);
  std::vector<long long> a(n + 1);
  for (int i = 1; i <= n; i++) {
    scanf("%I64d", &a[i]);
    a[i] += a[i - 1];
  }
  int ans = 0;
  for (int i = 0; i < (int) a.size(); i++) {
    for (int j = i + 1; j < (int) a.size(); j++) {
      if (a[j] > a[i]) {
        ans = std::max(ans, j - i);
      }
    }
  }
  printf("%d\n", ans);
}
```

Задача 4 – Выгодный бизнес – полное решение

Идеи решения: префиксные суммы, сортировка, метод двух указателей

1). Находим префиксные суммы.

2). Сортируем массив s, содержащий тройки <*преф_сумма*, *позиция*, *макс_позиция*>, по возрастанию преф-х сумм

3). Далее в массиве s для каждого элемента *s[i]* вычисляем значение поля *макс_позиция* – это будет максимальное значение поля "*позиция*", которое встречается в массиве s справа от *i* (включая *i*)

4). Используем метод двух указателей. Пусть i – левый указатель в массиве, *j* – правый указатель.
Сдвинув один раз вправо указатель *i*, теперь двигаем вправо указатель *j* столько раз, чтобы разность (s[j].sum - s[i].sum) стала больше нуля.
Теперь максимальную длину отрезка с такой (или большей) суммой даёт разность (s[j].макс_позиция – s[i].позиция).
Запоминаем наилучший вариант среди всех *i*.

Вычислительная сложность:
Пункты 1, 3, 4 – O(N), пункт 2 – O(n·log(N)).

Итого T(N) = O(n·log(N))

Решение задачи «Выгодный бизнес» на языке C++:

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

struct Elem {
  long long sum;
  int pos, maxpos;
  bool operator < (Elem s) const {
    return sum < s.sum || sum == s.sum && pos < s.pos;
  }
};

int main() {
  int n; scanf("%d", &n);
  std::vector<int> a(n + 1);
  std::vector<Elem> s(n + 1);
  for (int i = 1; i <= n; i++) {
    scanf("%d", &a[i]);
    s[i].sum = s[i - 1].sum + a[i];
    s[i].pos = i;
  }
  std::sort(s.begin(), s.end());
  s[s.size() - 1].maxpos = s[s.size() - 1].pos;
  for (int i = (int) s.size() - 2; i >= 0; i--)
    s[i].maxpos = std::max(s[i].pos, s[i + 1].maxpos);
  int ans = 0;
  for (int i = 0, j = i + 1; i < (int) s.size(); i++) {
    while (j < (int) s.size() && s[j].sum - s[i].sum <= 0)
      j++;
    if (j == (int) s.size()) break;
    ans = std::max(ans, s[j].maxpos - s[i].pos);
  }
  printf("%d\n", ans);
}
```

Задача 5 – Лягушка на полигоне – частичные решения

Подзадача 1 (*K* <= 20) - можно решить перебором всех путей (backtracking).

Подзадача 2 (*K* ≤ 10^4) - метод динамического программирования:
f[i] – количество способов попасть в вершину i. Сложность: O(*N*·*K*).

Решение на C++:

```
int n, k, ans = 0;
scanf("%d %d", &n, &k);
std::vector<int> f1(n + 1), f2(n + 1);
f1[1] = 1;
for (; k > 0; k--) {
  f2.assign(n  + 1, 0);
  for (int i = 1; i <= n; i++) {
    f2[i] += f1[i - 1 > 0 ? i - 1 : n];
    f2[i] += f1[i + 1 <= n ? i + 1 : 1];
    f2[i] %= 1000000007;
  }
  f1.swap(f2);
}
printf("%d\n", f1[1]);
```

Задача 5 – Лягушка на полигоне – полное решение

Пусть a - матрица смежности графа:
a[i][j] = 1, если есть ребро i - j, иначе a[i][j] = 0.

Как найти количество путей длины *K* от каждой вершины до каждой?
Для этого достаточно возвести матрицу a в *K*-ю степень
(см. здесь: http://e-maxx.ru/algo/fixed_length_paths)
Тогда $a^K[i][j]$ -количество путей длины K от *I* до j.

Но даже используя бинарное возведение в степень
(см. http://e-maxx.ru/algo/binary_pow), получаем сложность
$O(N^3 \cdot \log(K))$ - это слишком много. Что же делать?...

Заметим, что наш граф имеет специфический вид:
$a[i_1][j_1] = a[i_2][j_2]$, если расстояние по рёбрам полигона от вершины i_1 до j_1 равно расстоянию от i_2 до j_2.
То есть граф можно представлять не всей матрицей, а лишь её первой строкой.
Следствие: находить произведение таких однострочных «матриц» можно не за $O(N^3)$, а за $O(N^2)$.

Итого сложность алгоритма: $O(N^2 \cdot \log(K))$.

Решение задачи «Лягушка на полигоне» на языке C++:

```cpp
#include <stdio.h>
#include <vector>
#include <algorithm>
#include <cassert>

int n, k;
typedef std::vector<int> IntVect;
IntVect d1;

inline int dist(int from, int to) {
  if (from > to) std::swap(from, to);
  return std::min(to - from, n - to + from);
}

IntVect mult(const IntVect &a, const IntVect &b) {
  IntVect c(n / 2 + 1);
  for (int i = 0; i <= n / 2; i++) {
    for (int mid = 0; mid < n; mid++) {
      int d1 = dist(0, mid);
      int d2 = dist(mid, i);
      c[i]=(c[i]+1LL*a[d1]*b[d2])%1000000007;
    }
  }
  return c;
}
```

```cpp
IntVect pow(const IntVect &a, int k) {
  if (k == 1) return d1;
  if (k % 2 == 1) {
    return mult(pow(a, k - 1), a);
  } else {
    IntVect b = pow(a, k / 2);
    return mult(b, b);
  }
}

int main() {
  scanf("%d %d", &n, &k);
  d1.resize(n / 2 + 1);
  d1[1] = 1;
  printf("%d\n", pow(d1, k)[0]);
}
```

Литература и web-ресурсы для подготовки

1. Дистанционный практикум по программированию ВоГУ: http://atpp.vstu.edu.ru/acm
2. Школа программиста: http://acmp.ru/
3. Алгоритмы на e-maxx: http://e-maxx.ru/algo/
4. Базовые алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/997/313/info
5. Базовые и "продвинутые" алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/998/312/info
6. "Продвинутые" алгоритмы для школьников" (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/975/311/info
7. Олимпиадное программирование с нуля на Java: https://vk.com/ol_prog_0
8. Проект "3.5 задачи в неделю“: http://codeforces.com/blog/entry/20066
9. Соревнования по программированию на Codeforces: http://codeforces.com
10. Шень, А. “Программирование: теоремы и задачи”: http://www.e-academy7.narod.ru/COURSES/PROGRAM/LITERATURA/01shen.PDF
11. Меньшиков, Ф. В. Олимпиадные задачи по программированию/ Меньшиков, Федор Владимирович. - Москва: Питер, 2006. - 315 с.
12. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В. М. Гуровица и В. А. Матюхина—М.: МЦНМО, 2006.— 256 с
13. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен , Ч. Лейзерсон, Р. Ривест, К. Штейн; пер. с англ.; 3-е изд. - Москва: ООО "И.Д. "Вильямс", 2013. - 1328 с.
14. Скиена С.С., Ревилла М.А. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. – М.: Кудиц-образ, 2005. – 416 с.

И многое-многое другое…

Электронные версии книг можно поискать на http://gen.lib.rus.ec