Алгоритмы и структуры данных 1 модуль.

Андрей Тищенко

2024/2025

Семинар 3 сентября.

Оценка за работу на семинаре

Посещаемость и ответы у доски. Каждое посещение это 0.5 балла, ответ у доски - 1 балл (максимум 10 баллов). Задача про определение конечности списка. (рисуночки, идеи)
Задача: написать очередь через два стека.

```
struct queue {
    stack st1, st2;
    void push(int x);
    void pop();
    int front();
    void transition();
};
void push(int x) {
    st1.push(x);
}
void pop() {
    if (st2.empty()) {
        transition();
    }
    st2.pop();
}
void transition() {
    while (st1.size()) {
        st2.push(st1.top());
        st1.pop();
    }
}
```

Семинар 10 сентября

На лекции не разобрали случай $c > \log_b a$. Под этот случай подходит алгоритм Карацубы, но его разберём на следующей лекции. Вопрос в лекции про $k \log k = n \Rightarrow k = O(?)$. $k = O\left(\frac{n}{\log n}\right)$, подставим это:
$\frac{n}{\log n} \log \frac{n}{\log n} = \frac{n}{\log n}(\log |n| - \log \log n)$
$\log |n| - \log \log n = O(\log n) \Rightarrow \frac{n}{\log n}(\log |n| - \log \log n) = n\frac{O(\log n)}{\log n} = nO(1) =$
$= O(n)$

Задача 1

Найти подотрезок с заданной суммой S. Разделяем отрезок на две половины. Ответ может лежать в одной из половин или на из пересечении, тогда нужно сохранить в map все суффиксы левого отрезка, просмотреть префиксы правого отрезка, для отрезков длины sum нас интересует существование суффикса $S - sum$.
То есть формируем map M и смотрим $M.count(S - sum) == 1$
$T(n) = 2T\left(\frac{n}{2}\right) + O\left(n\right) \Rightarrow T(n) = n \log n$ если отрезок полностью внутри одной половины. Иначе будет $T(n) = 2T\left(\frac{n}{2}\right) + O\left(n \log n\right) \Rightarrow$
$\Rightarrow T(n) = n \log^2 n$.

Задача 2

Даны n точек, нужно найти две точки, расстояние между которыми минимально. Должно работать бытсрее $O(n^2)$.
Сделаем поворот плоскости таким образом, чтобы ни одна пара точек не лежала на одной вертикальной прямой. Проведём одну вертикальную прямую так, чтобы слева и справа от неё было поровну точек и эта прямая не пересекает ни одну точку. Пусть кратчайшее растояние между точками слева d_1, а справа - d_2.
Тогда обозначим $d = min(d_1,\ d_2)$. Проведём две параллельные прямые слева с справа от изначальной прямой на расстоянии d. Поделим образовавшуюся линию на кваратики со сторонами $\frac{d}{2}$, в каждом таком квадрате не больше одной точки, так как иначе расстояние между ними было бы меньше d, но мы рекуррентно поняли, что в любой половине минимальное расстояние d.
Тогда останется посмотреть не более 11 квадратов для каждой рассматриваемой точки. Однако для такого просмотра нужно отсортировать точки по высоте. На рекуррентных вызовах тоже нужно будет это сортировать, поэтому можно mergить результаты из двух половин. Тогда ассимптотика

алгоритма будет:
$T(n) = 2T(\frac{n}{2}) + 11n = O(n \log n)$
Если на каджом шаге делать сортировку заново, то ассимптотика будет:
$T(n) = 2T(\frac{n}{2}) + 11n + n \log n = O(n \log^2 n)$, то есть медленнее.

Задача 3

Дано несколько точек, нужно упорядочить их в полярных координатах.
$(v_i \times v_j) > 0 \Leftrightarrow i < j$
Но может быть проблема: $\exists i,\ j,\ k:\ i < j < k < i$

Семинар 17 сентября

```
#include <vector>

// Multiplying vectors of the same size
std::vector<long long> bMult(std::vector<long long> a,
    std::vector<long long> b) {
    int n = a.size();
    std::vector<long long> c(2*n, 0);
    for (int k = 0; k < 2*n; ++k) {
        for (int i = std::max(0, k - n + 1); i <= k && i < n;
            ++i) {
            c[k] += a[i]*b[k - i];
        }
    }
    return c;
}

std::vector<int> kar(std::vector<long long> a,
    std::vector<long long> b) {
    int n = a.size();
    int nHalf = n/2;
    if (n <= 32) {
        return bMult(a, b);
    }
    std::vector<long long> a0(nHalf, 0), a1(nHalf, 0),
        b0(nHalf, 0), b1(nHalf, 0);
    // a0(a.begin(), a.begin() + nHalf) - fill with iterators
    std::vector<long long> a0b0, a1b1, abab;
    // simple fill
    for (int i = 0; i < nHalf; ++i) {
        a0[i] = a[i];
```

```
            a1[i] = a[i + nHalf];
            b0[i] = b[i];
            b1[i] = b[i + nHalf];
        }
        a0b0 = kar(a0, b0);
        a1b1 = kar(a1, b1);
        // a0 will become a0 + a1 and b0 will become b0 + b1
        for (int i = 0; i < nHalf; ++i) {
            a0[i] += a1[i];
            b0[i] += b1[i];
        }
        abab = kar(a0, b0);
        abab -= (a0b0 + a1b1);
        std::vector<long long> ans(2*n, 0);
        for (int i = 0; i < n; ++i) {
            ans[i] += a0b0[i];
            ans[i + n/2] += abab[i];
            ans[i + n] += a1b1[i];
        }
        return ans;
    }
```

Пример длинной арифметики:
$x = 10: \quad 123 \cdot 225 = (3+2x+x^2)\cdot(5+2x+2x^2) = 15+16x+15x^2+6x^3+2x^4$
В многочленах ответ получили, перевод в число в векторах будет выглядеть примерно так:
$\boxed{15 \mid 16 \mid 15 \mid 6 \mid 2 \mid 0 \mid 0 \mid 0} \Rightarrow \boxed{5 \mid 7 \mid 6 \mid 7 \mid 2 \mid 0 \mid 0 \mid 0}$
$x = 100: \quad 1001 \cdot 1002 = 1003002 = (10x+1)(10x+2) = 100x^2 + 30x + 2$
$\boxed{2 \mid 30 \mid 100 \mid 0} \Rightarrow \boxed{2 \mid 30 \mid 0 \mid 1}$. Но если это восстановить, получаем 10302, но это неправда, ведь 2 и 0, которые хранятся в векторе по факту являются 02 и 00:
$\boxed{02 \mid 30 \mid 00 \mid 1} \Rightarrow 1003002$

Задача

Даются числа $a_1, a_2, \dots, a_n$ и $b_1, b_2, \dots, b_m$.
Известно, что $0 \leqslant a_i, b_i \leqslant 10^5$, $n, m \leqslant 10^5$.
$\forall k \in \{0, 1, \dots, 2 \cdot 10^5\}, \ num[k] = \#\{(i, j) : a_i + b_j == k\}$
$A(x) = x^{a_1} + x^{a_2} + \dots + x^{a_n}$
$B(x) = x^{b_1} + x^{b_2} + \dots + x^{b_m}$
$C(x) = A(x) \cdot B(x) = c_0 + c_1 x + \dots + \underbrace{c_k}_{=num[k]} x^k + \dots$

$x^{a_i} \cdot x^{b_j} = x^k \Rightarrow c_k = \#\{(i,\ j) : a_i + b_j == k\}$, что нам и нужно.
Задача считается сложной и может появиться на коллоквиуме как вопрос на 9, 10 баллов.

Алгоритм Штрассена

$a_{11}b_{11} + a_{12}b_{21} = d + d_1 + v_1 - h_1$
$a_{11}b_{12} + a_{12}b_{22} = h_1 + v_2$
$a_{21}b_{11} + a_{22}b_{21} = h_2 + v_1$
$a_{21}b_{12} + a_{22}b_{22} = d + d_2 + v_2 - h_2$
$d = (a_{11} + a_{22})(b_{11} + b_{22})$
$d_1 = (a_{12} - a_{22})(b_{21} + b_{22})$
$d_2 = (a_{21} - a_{11})(b_{11} + b_{12})$
$h_1 = (a_{11} + a_{12})b_{22}$
$h_2 = (a_{21} + a_{22})b_{11}$
$v_1 = a_{22}(b_{21} - b_{11})$
$v_2 = a_{11}(b_{12} - b_{22})$

Семинар 24 сентября.

Задача 1

Пусть есть функция $rnd2()$, которая равновероятно возвращает 0 или 1.
а. На её основе написать $rnd4()$ и $rnd8()$

```
int rnd2() {...}
int rnd4() {
    return rnd2() + 2*rnd2();
}
int rnd8() {
    return rnd2() + 2*rnd2() + 4*rnd2();
}
```

б. На её основе написать $rnd3()$.

```
int rnd2() {...}
int rnd3() {
    int x[3] = {rnd2(), rnd2(), rnd2()};
    while (x[0] + x[1] + x[2] != 1) {
        x[0] = rnd2();
        x[1] = rnd2();
        x[2] = rnd2();
    }
```

```
    for (int i = 0; i < 3; i++) {
        if (x[i]) {
            return i;
        }
    }

}

int rnd3_author() {
    int x = 3;
    while(x == 3) {
        x = rnd(4);
    }
    return x;
}
```

Вероятность неудачи $rnd3() = \frac{5}{8}$, у $rnd3\_author() = \frac{1}{4}$.
Матожидание количества вызовов $rnd4()$ из $rnd3()$:
$E = \frac{3}{4} \cdot 1 + \frac{1}{4}(1 + E) \Rightarrow \frac{3}{4}E = 1 \Rightarrow E = \frac{4}{3}$

Задача 2

Найти наименьшую обхатывающую окружность для n точек.
Можно решить вероятностным алгоритмом за ожидаемое время $O(n)$.
Выберем случайную точку, построим для неё оболочку (окружность радиуса 0), добавляем ещё одну, строим обхатывающую для пары точек.
Добвляем ещё одну точку. Возможны следующие исходы:

1. Точка попала в круг
2. Точка не попала в круг

Утвреждение (без доказательства, оно сложное): точка, не попавшая в круг должна быть на границе новой окружность.

```
class Point {...};
class Circle {... bool owns(Point) ...};

Circle ansCircle = ...;

Circle minD(std::vector<Point> x) {
    k = x.size();
    if (ansCircle.owns(x[k - 1])) {
        return minD(x.pop_back());
```

```
    }
    return minD1(x[k - 1], x.pop_back())
}
Circle minD1(Point y, std::vector<Point> x) {
    k = x.size();
    Circle c = mind1(y, x.pop_back());
    if (c.owns(x[k - 1])) {
        return c;
    }
    return minD2(y, x[k - 1], x.pop_back());
}

Circle minD2(Point y1, Point y2, std::vector<Point> x) {
    k = x.size();
    if (!k) {
        return Circle(y1, y2);
    }
    c = minD2(y1, y2, x.pop_back());
    if (c.owns(x[k - 1])) {
        return c;
    }
    return circle(x[k - 1], y1, y2);
}
```

Матожидание minD2: $E\big(T_2(k)\big) = T_2(k-1) + \frac{k-1}{k} \cdot 0 + \frac{1}{k}1 = O(k)$
Матожидание minD1: $E\big(T_1(k)\big) = E\big(T_1(k-1)\big) + \frac{k-1}{k} \cdot 0 + \frac{1}{k}\underbrace{T_2(k-1)}_{O(k-1)} =$

$= T_1(k-1) + \frac{O(k-1)}{k} = T_1(k-1) + O(1)$