

SPb HSE, 1 курс, осень 2021/22

Практика по алгоритмам #2

Асимптотика, начало структур

11 сентября

Собрано 10 сентября 2021 г. в 19:43

Содержание

1. Асимптотика, начало структур	1
1.1. Стандартные задачи на стек	1
1.2. Неасимптотические оптимизации	1
1.3. Задачи про цикл for	2
1.4. Дополнительные задачи	3
2. Разбор задач практики	4
2.1. Стандартные задачи на стек	4
2.2. Неасимптотические оптимизации	4
2.3. Задачи про цикл for	6
2.4. Дополнительные задачи	7
3. Домашнее задание	8
3.1. Обязательная часть	8
3.2. Дополнительная часть	10

Асимптотика, начало структур

1.1. Стандартные задачи на стек

1. Проверить правильность скобочной последовательности с несколькими типами скобок.
2. Найти значение арифметического выражения, содержащего числа и символы $+ - * () ^$.
3. Даны два выражения с целыми числами и операциями $+$, $-$, $*$.
Суммарная длина не превосходит 10^6 . Проверить, равны ли значения выражений.

1.2. Неасимптотические оптимизации

1. Функция go

```
1 void go(int n) {
2     if (n <= 0) return;
3     a[k++] = n;
4     go(n - 1);
5     a[k++] = n;
6 }
```

2. Функция projection

```
1 void projection(int n, int *x, int *y, double *z, double angle) {
2     for (int i = 0; i < n; i++)
3         z[i] = x[i] * cos(angle) + y[i] * sin(angle); // проекция
4 }
```

3. Цикл for

```
1 int n, p[n], a[n], b[n]; // p - перестановка
2 for (int k = 0; k < n; k++) {
3     int cnt = 0;
4     for (int i = k; i < n - k; i++)
5         if (a[p[i]] >= b[k])
6             cnt++;
7     printf("%d\n", cnt);
8 }
```

4. Цикл for с вектором

```
1 for (int i = 1; i < n; i++) {
2     vector<int> digits;
3     for (int j = i, d = 2; j > 0; j /= d, d++)
4         digits.push_back(j % d);
5     for (int j = digits.size() - 1; j >= 0; j--)
6         printf("%d ", digits[j]);
7     printf("\n");
8 }
```

1.3. Задачи про цикл for

Разминочная задача: ищем такие a, b, c : $ab + bc + ca = N$.

Оцените сложность фрагментов программ.

1. Ищем такие a, b, c : $abc = N$, $a + b + c = \min$.

```
1 for (int a = 1; a <= N; a++)
2   for (int b = 1; a * b <= N; b++)
3     { int c = N / a / b, ... }
```

2. Ищем такие a, b, c : $abc = N$, $a + b + c = \min$.

```
1 for (int a = 1; a * a * a <= N; a++)
2   for (int b = 1; b * b <= N; b++)
3     { int c = N / a / b, ... }
```

3. Поиск делителей без деления

```
1 int b = N;
2 for (int a = 1; a <= N; a++) {
3   while (a * b > N)
4     b--;
5   if (a * b == N)
6     printf("%d %d\n", a, b);
7 }
```

4. Повторение, мать!

```
1 for (int a = 1; a < n; a++)
2   for (int b = 0; b < n; b += a)
3     ;
```

5. Partition

```
1 int a = 1, b = n, M = x[n / 2];
2 while (a < b) {
3   while (x[a] < M) a++;
4   while (x[b] > M) b--;
5   if (a <= b) swap(x[a++], x[b--]);
6 }
```

Что делает этот код?

6. Перестановки и циклы

```
1 for (int i = 1; i < n; i++)
2   if (used[i] == 0)
3     for (int j = i; used[j] == 0; j = (j * 17 + 2) % n)
4       used[j] = 1;
```

7. Дежавю

```
1 int y = N, sum_x = 0, sum_y = 0;
2 for (int x = 1; x <= N; x++) {
3   while (x * y > N) y--;
4   sum_x += x, sum_y += y;
5 }
```

Какова асимптотика sum_x и sum_y после выполнения программы?

1.4. Дополнительные задачи

8. Sqrt*

Сколько работает этот код?

```
1 while (N > 2)
2   N = sqrt(N)
```

А если бы вместо 2 было 1?

9. Ищем такие a, b, c : $abc = N$, $a + b + c = \min$.

```
1 for (int a = 1; a * a * a <= N; a++)
2   for (int b = a; a * b * b <= N; b++)
3     { int c = N / a / b, ... }
```

10. Решето Эратосфена

```
1 for (int p = 2; p < n; p++)
2   if (min_divisor[p] == 0) // число p простое
3     for (int x = p + p; x < n; x += p)
4       if (min_divisor[x] == 0)
5         min_divisor[x] = p;
```

11. Странный код

```
1 int a[k][n + 1]; // a[i=0..k-1][n] = -infty
2 int p[k]; // p[i=0..k-1] = 0
3 for (int i = 0; i < m; i++) { // m <= n * k
4   int min_j = 0;
5   for (int j = 0; j < k; j++)
6     while (a[j][p[j]] < a[min_j][p[min_j]])
7       min_j = j;
8   for (int j = 0; j < k; j++)
9     while (a[j][p[j]] > a[min_j][p[min_j]])
10      p[j]++;
11 }
```

12. Выражение, тождественно равное нулю

Дано выражение с целыми числами и операциями $+$, $-$, $*$. А также переменными x_1, x_2, x_3, \dots . Суммарная длина не превосходит 10^6 .

Проверить, является ли выражение тождественным нулём.

Разбор задач практики

2.1. Стандартные задачи на стек

1. Скобки.

Открывающие скобки кладем в стек. Встретив закрывающую, сравним ее тип с открывающей на вершине стека и сделаем pop. Плохо, если не сошелся тип, либо если стек в конце не пуст.

Индуктивное предположение для обоснования корректности: S – ПСП \Leftrightarrow после обработки алгоритмом строки S стек остался в том же состоянии, что до обработки S .

2. Значение арифметического выражения

Два стека `numbers` и `operators`. «Применить оператор» значит вынуть y, x из `numbers` и \circ из `operators`, затем `numbers.push(x \circ y)`.

Идем по выражению слева направо. Встречая число x , кладем в `numbers`. Встречая оператор \circ , пока приоритет `operators.top()` выше или равен, применяем `operators.top()`, в конце кладем \circ в `operators`.

Открывающие скобки кладем в `operators`. Встречая закрывающую, применяем верхний оператор, пока он не открывающая скобка.

Еще надо выполнить все операторы, оставшиеся в стеке после всего прохода по выражению (вместо этого можно заключить все выражение в скобки). Окончательный результат будет на вершине `numbers`.

Описанная схема не работает для \wedge : $a \wedge b \wedge c = a^{b^c}$, а не $(a^b)^c$. Для \wedge надо снимать со стека `operators` только более высокий приоритет, но не равный.

Дело в том, что \wedge – правоассоциативная операция, а не левоассоциативная.

3. Проверка равенства двух длинных выражений

Если вычислять значения явно, могут получиться числа длины $\approx 10^6$, операции с ними долгие. Проверим, что их значения равны по модулю $P = 10^9 + 7$. Для большей верности можно проверить по модулю нескольких случайных простых чисел в пределах $2 \cdot 10^9$.

Вероятность того, что выражения не равны, а остатки по модулю случайного числа из $[2, P]$ равны, $\leq \frac{10^6}{P}$, так как у их разности не более 10^6 простых делителей.

2.2. Неасимптотические оптимизации

1. `void go(int n)`. Вывод – массив $n, n-1, \dots, 2, 1, 1, 2, \dots, n-1, n$. Цикл быстрее рекурсии.

2. **Проекция.** Сохранить в переменные `cos(angle)`, `sin(angle)`, а не считать n раз заново.

3. **Перестановка.** `for (int i = 0; i < n; ++i) ap[i] = a[p[i]]` в начало. Мы n раз проходили `a` не по порядку, кэш грустил.

Также можно ускорить вывод.

4. **Разложение всех i .** Объявить `digits` вне цикла и делать в цикле `digits.clear()`. Мы n раз вызвали его конструктор и деструктор. Также можно ускорить вывод.

2.3. Задачи про цикл for

Разминка:

- можно перебрать for $a=1..n$, for $b=1..n$, for $c=1..n$. $\mathcal{O}(n^3)$.
 - можно перебрать for $a=1..n$, for $b=1..n$. $\mathcal{O}(n^2)$.
 - можно перебрать for $a=1..n$, for $b=1..n/a$. $\mathcal{O}(n \log n)$.
 - можно понять, что $a \leq b \leq c \Rightarrow b \leq \sqrt{n}$ перебрать for $a=1..\sqrt{n}$, for $b=1..\sqrt{n}$. $\mathcal{O}(n)$.
1. **Ищем такие a, b, c :** $abc = N$, $a + b + c = \min$. Для каждого a прошли от 1 до $\lfloor \frac{N}{a} \rfloor$, итого $N(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}) = \Theta(N \log N)$.
 2. **Ищем такие a, b, c :** $abc = N$, $a + b + c = \min$. $\Theta(N^{1/3} \cdot N^{1/2}) = \Theta(N^{5/6})$.
 3. **Поиск делителей без деления.** $\Theta(N)$. Время = N плюс суммарное время всех while. Всегда $b > 0$, при $a = N$ станет $b = 1$, значит, не более N раз будет b--.
 4. **Повторение, мать!** Снова $\sum_{a=1}^n \frac{n}{a} = \Theta(n \log n)$.
 5. **Partition.** $\Theta(n)$. Указатель a пройдет от 1 до не более n , так как всегда впереди него есть M . Если a наткнется на M , то переносит его вперед. Аналогично b . В итоге в массиве сначала будут элементы $\leq M$, затем $\geq M$.
 6. **Перестановки и циклы.** $\Theta(n)$, для каждого j ровно один раз делается $\text{used}[j] = 1$.
 7. **Дежавю.** Время $\Theta(N)$. $x = 1 + 2 + \dots + N = \Theta(N^2)$. $y = \sum \frac{1}{x} = \Theta(N \log N)$.

2.4. Дополнительные задачи

8. Sqrt*

На каждом шаге $\log N$ уменьшается вдвое, итого $\Theta(\log \log N)$.

Если 2 заменить на 1, то в теории выйдет бесконечно долго, но на практике цикл завершается по достижении $1 + \varepsilon$, где $\varepsilon = 10^{-15}$ для типа `double`. $\sqrt{1+x} = 1 + \frac{x}{2} + o(x)$ при $x \rightarrow 0$, поэтому время работы $\Theta(\log \log N + \log \frac{1}{\varepsilon})$. Тестирование:

```

1 int main() { // g++ -O0
2   int k = 0;
3   for (double x = 1e9; x > 1; x = sqrt(x))
4     k++;
5   printf("%d\n", k); // 57
6 }

```

9. Ищем такие a, b, c : $abc = N$, $a + b + c = \min$.

$$\sum_{a=1}^{N^{1/3}} \sqrt{\frac{N}{a}} = \Theta\left(\int_1^{N^{1/3}} \sqrt{\frac{N}{a}} da\right) = \Theta(N^{1/2} a^{1/2} \Big|_1^{N^{1/3}}) = \Theta(N^{1/2+1/6}) = \Theta(N^{2/3}).$$

10. Решето Эратосфена. Время $\Theta(n \log \log n)$.

Магический факт: $p_k = \Theta(k \ln k)$.

$$\text{Тогда сложность } \sum_{k=2}^{n/\ln n} \Theta\left(\frac{n}{k \ln k}\right) = n \cdot \Theta\left(\int_2^{n/\ln n} \frac{dx}{x \ln x}\right) = n \cdot \Theta(\ln \ln x \Big|_2^{n/\ln n}) = \Theta(n \ln \ln n).$$

11. Странный код

Каждый указатель p_i делает не более n шагов вперёд, так как последний столбец $-\infty$. Еще можно заметить, что первый `while` каждый раз делает не более одного шага. Итого $\mathcal{O}(k(n+m))$.

12. Выражение, тождественно равное нулю

Несколько раз подставим случайные числа в переменные и проверим, ноль ли. Вычисляем по модулю $P = 10^9 + 7$.

Выражение из `+*()` – это многочлен, если он не нулевой, то у него не более его степени корней (по модулю P тоже). Вероятность попасть в корень равна $\frac{\text{deg}}{P}$.

Домашнее задание

3.1. Обязательная часть

1. (4) Цикл for

Оцените сложность фрагмента программы.

a) Два фора

```
1 for (int i = 1; i < n; i = i + i)
2   for (int j = 0; j < i; j++)
3     ;
```

b) Ещё два фора

```
1 for (int i = 0; i < n; i += 3)
2   for (int j = 1; j < i; j += j)
3     ;
```

c) И ещё два фора

```
1 for (int i = 1; i < n; i++)
2   for (int j = n/i; j < i; j++)
3     ;
```

d) Опять два фора

```
1 int x = n;
2 for (int i = n; i >= 1; i--)
3   for (; x > i; x--)
4     ;
```

2. (4.5) Неасимптотические оптимизации

Ускорьте код. Не обязательно писать новый код, укажите, что и как оптимизировать.

Не нужно менять сам алгоритм, сделайте чисто техническую оптимизацию.

Обязательно поясните, почему код был медленным и что стало лучше.

a) Скалярное произведение

```
1 const int MOD = 1e9; // 0 <= a[i], b[i] < MOD
2 long long scalarProductMod(int n, int64_t *a, int64_t *b) {
3   long long sum = 0;
4   for (int i = 0; i < n; i++) sum = (sum + a[i] * b[i]) % MOD;
5   return sum;
6 }
```

b) Степень ужаса

```
1 double Exp(double x, int dep=0, double F=1) {
2   if (dep >= 20) return 0;
3   return pow(x, dep) / F + Exp(x, dep+1, F*(dep+1));
4 }
5 double result = Exp(0.5); // пример использования
```

с) В этом пункте просто укажите медленные части.

`string` \approx `vector<char>`, `to_string`: `int` \rightarrow `string`.

```
1 void outputNatural(int n) {
2     string buf;
3     for (int i = 1; i <= n; i++) buf += to_string(i) + " ";
4     cout << buf << endl;
5 }
```

3. (3) Подотрезок с заданной суммой

Дана последовательность $a_1, a_2, \dots, a_n \in \mathbb{N}$ и $S \in \mathbb{N}$.

Найти l, r ($1 \leq l \leq r \leq n$) такие, что сумма $\sum_{i=l}^r a_i = S$. $\mathcal{O}(n)$. Частичный балл (1.5) за $\mathcal{O}(n^2)$.

3.2. Дополнительная часть

1. (1.5) Суммы и интегралы

Можно пользоваться интегралами (см. конспект), есть короткое решение без них.

(а) Докажите, что $\sum_{k=1}^n \frac{1}{k^2} = \mathcal{O}(1)$ (б) Оцените сумму $\sum_{k=1}^n \frac{1}{k^{1/2}}$

(с) Докажите, что $\sum_{k=1}^{\infty} \frac{1}{k^{3/2}} = \mathcal{O}(1)$

2. (3) Частый элемент

Дана последовательность объектов a_1, a_2, \dots, a_n . Над объектами определена операция сравнения на равенство. Известно, что в последовательности есть элемент присутствующий строго больше, чем $\frac{n}{2}$ раз. Требуется найти элемент a за линейное $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти.

3. (2) Подотрезок с заданной суммой

Дана последовательность $a_1, a_2, \dots, a_n \in \mathbb{Z}$ и $S \in \mathbb{Z}$.

Найти l, r ($1 \leq l \leq r \leq n$) такие, что сумма $\sum_{i=l}^r a_i = S$. $\mathcal{O}(n)$.

P.S. В этой задаче можно пользоваться тем, чего мы ещё не проходили.

4. (3) Делители

Пусть $d(n)$ – количество делителей числа n . Докажите $\forall \varepsilon > 0, d(n) = o(n^\varepsilon)$.

P.S. Более точно $d(n) \approx n^{1/\log \log n}$: https://en.wikipedia.org/wiki/Divisor_function.