Первый курс, осенний семестр 2017/18
Практика по алгоритмам #2

Асимптотика, начало структур

15 сентября

Собрано 15 сентября 2018 г. в 13:22

Содержание

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. Оптимизации

1. ```
   void go( int n ) {
     if (n <= 0)
       return;
     a[k++] = n;
     go(n - 1);
     a[k++] = n;
   }
   ```
2. ```
   void projection( int n, int *x, int *y, double *z, double angle ) {
     for (int i = 0; i < n; i++)
       z[i] = x[i] * cos(angle) + y[i] * sin(angle); // проекция
   }
   ```
3. ```
   int n, p[n], a[n], b[n]; // p - перестановка
   for (int k = 0; k < n; k++) {
     int cnt = 0;
     for (int i = k; i < n - k; i++)
       if (a[p[i]] >= b[k])
         cnt++;
     printf("%d\n", cnt);
   }
   ```
4. ```
   for (int i = 1; i < n; i++) {
     vector<int> digits;
     for (int j = i, d = 2; j > 0; j /= d, d++)
       digits.push_back(j % d);
     for (int j = digits.size() - 1; j >= 0; j--)
       printf("%d ", digits[j]);
     printf("\n");
   }
   ```

1.3. Задачи про цикл for

Оцените сложность фрагмента программы.

1. **Ищем такие** a, b, c: $abc = N$, $a + b + c = \min$.
```
for (int a = 1; a <= N; a++)
  for (int b = 1; a * b <= N; b++)
  {  int c = N / a / b, ... }
```
2. **Ищем такие** a, b, c: $abc = N$, $a + b + c = \min$.
```
for (int a = 1; a * a * a <= N; a++)
  for (int b = 1; b * b <= N; b++)
  {  int c = N / a / b, ... }
```
3. **Поиск делителей без деления**
```
int b = N;
for (int a = 1; a <= N; a++) {
  while (a * b > N)
    b--;
  if (a * b == N)
    printf("%d %d\n", a, b);
}
```
4. **Повторение, мать!**
```
for (int a = 1; a < n; a++)
  for (int b = 0; b < n; b += a)
    ;
```
5. **Partition**
```
int a = 1, b = n, M = x[n / 2];
while (a < b) {
  while (x[a] < M) a++;
  while (x[b] > M) b--;
  if (a <= b) swap(x[a++], x[b--]);
}
```
Что делает этот код?

6. **Перестановки и циклы**
```
for (int i = 1; i < n; i++)
  if (used[i] == 0)
    for (int j = i; used[j] == 0; j = (j * 17 + 2) % n)
      used[j] = 1;
```
7. **Дежавю**
```
int y = N, sum_x = 0, sum_y = 0;
for (int x = 1; x <= N; x++) {
  while (x * y > N)
    y--;
  sum_x += x, sum_y += y;
}
```
Какова асимптотика `sum_x` и `sum_y` после выполнения программы?

1.4. Дополнительные задачи

8. **Sqrt***
   Сколько работает этот код?

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

   А если бы вместо 2 было 1?

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

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

10. **Решето Эратосфена**

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

11. **Странный код**

```
int a[k][n + 1]; // a[i=0..k-1][n] = -infty
int p[k]; // p[i=0..k-1] = 0
for (int i = 0; i < m; i++) { // m <= n * k
  int min_j = 0;
  for (int j = 0; j < k; j++)
    while (a[j][p[j]] < a[min_j][p[min_j]])
      min_j = j;
  for (int j = 0; j < k; j++)
    while (a[j][p[j]] > a[min_j][p[min_j]])
      p[j]++;
}
```

12. **Выражение, тождественно равное нулю**
    Дано выражение с целыми числами и операциями `+`, `-`, `*`. А также переменными $x_1, x_2, x_3, \ldots$.
    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 `^`: $a \hat{} b \hat{} c = a^{b^c}$, а не $(a^b)^c$. Для `^` надо снимать со стека `operators` только более высокий приоритет, но не равный.
   Дело в том, что `^` – 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, $\leqslant \frac{10^6}{P}$, так как у их разности не более 10^6 делителей.

2.2. Оптимизации

1. `void go(int n)`. Вывод – массив $n, n-1, \ldots, 2, 1, 1, 2, \ldots, n-1, n$. 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

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 $N(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}) = \Theta(N \log N)$.
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 $\sum\limits_{a=1}^{n} \frac{n}{a} = \Theta(n \log n)$.
5. **Partition.** $\Theta(n)$. Указатель a пройдет от 1 до не более n, так как всегда впереди него есть M. Если a натыкается на M, то переносит его вперед. Аналогично b. В итоге в массиве сначала будут элементы $\leqslant M$, затем $\geqslant M$.
6. **Перестановки и циклы.** $\Theta(n)$, для каждого j ровно один раз сделается `used`$[j] = 1$.
7. **Дежавю.** Время $\Theta(N)$. $x = 1 + 2 + \ldots + N = \Theta(N^2)$. $y = \sum \frac{1}{x} = \Theta(N \log N)$.

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 бесконечо долго, но на пратике цикл завершается по достижении $1+\varepsilon$, где $\varepsilon = 10^{-15}$ для типа `double`. $\sqrt{1+x} = 1 + \frac{x}{2} + o(x)$ при $x \to 0$, поэтому время работы $\Theta(\log \log N + \log \frac{1}{\varepsilon})$. Тестирование:
   ```
   int main() { // g++ -O0
     int k = 0;
     for (double x = 1e9; x > 1; x = sqrt(x))
       k++;
     printf("%d\n", k); // 57
   }
   ```
9. **Ищем такие** a, b, c: $abc = N$, $a + b + c = \min$.
   $\sum_{a=1}^{N^{1/3}} \sqrt{\frac{N}{a}} = \Theta(\int_1^{N^{1/3}} \sqrt{\frac{N}{a}} da) = \Theta(N^{1/2} a^{1/2} |_1^{N^{1/3}}) = \Theta(N^{1/2+1/6}) = \Theta(N^{2/3})$.
10. **Решето Эратосфена.** Время $\Theta(n \log \log n)$.
    Магический факт: $p_k = \Theta(k \ln k)$.
    Тогда сложность $\sum\limits_{k=2}^{n/\ln n} \Theta(\frac{n}{k \ln k}) = n \cdot \Theta(\int\limits_2^{n/\ln n} \frac{dx}{x \ln x}) = n \cdot \Theta(\ln \ln x |_2^{n/\ln n}) = \Theta(n \ln \ln n)$.
11. **Странный код**
    Каждый указатель p_i сделает 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 $\frac{\deg}{P}$.

Домашнее задание

3.1. Обязательная часть

1. **(4) Суммы и интегралы**
   Можно пользоваться интегралами (см.конспект), есть короткое решение без них.
   (a) Докажите, что $\sum_{k=1}^{n} \frac{1}{k^2} = \mathcal{O}(1)$ (b) Оцените сумму $\sum_{k=1}^{n} \frac{1}{k^{1/2}}$.

2. **(2) Цикл for**
   Оцените сложность фрагмента программы.
   a)
   ```
   for (int i = 1; i < n; i = i + i)
     for (int j = 0; j < i; j++)
       ;
   ```
   b)
   ```
   for (int i = 0; i < n; i += 3)
     for (int j = 1; j < i; j += j)
       ;
   ```

3. **(4.5) Оптимизация**
   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)
   ```
   const int MOD = 1e9; // 0 <= a[i], b[i] < MOD
   long long scalarProductMod( int n, long long *a, long long *b ) {
     long long sum = 0;
     for (int i = 0; i < n; i++) sum = (sum + a[i] * b[i]) % MOD;
     return sum;
   }
   ```
   b)
   ```
   double dzeta( double x, int dep ) {
     if (dep >= 20) return 0;
     return pow(x, -dep) + dzeta(x, dep + 1);
   }
   double result = dzeta(0.5, 0);
   ```
   c) В этом пункте просто укажите медленные части.
   `string` $\approx$ `vector<char>`, `to_string: int` $\to$ `string`.
   ```
   void outputNatural( int n ) {
     string buf;
     for (int i = 1; i <= n; i++) buf += to_string(i) + " ";
     cout << buf << endl;
   }
   ```

4. **(3) Подотрезок с заданной суммой**
   Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$ и $S \in \mathbb{N}$.
   Найти l, r $(1 \le l \le r \le n)$ такие, что сумма $\sum_{i=l}^{r} a_i = S$. $\mathcal{O}(n)$. Частичный балл за $\mathcal{O}(n^2)$.

3.2. Дополнительная часть

1. **(2) Подотрезок с заданной суммой**
   Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{Z}$ и $S \in \mathbb{Z}$.
   Найти l, r $(1 \le l \le r \le n)$ такие, что сумма $\sum_{i=l}^{r} a_i = S$. $\mathcal{O}(n)$.

2. **(3) Частый элемент**
   Дана последовательность объектов $a_1, a_2, \ldots, a_n$. 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. **(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.