РТУ МИРЭА
Институт кибербезопасности и цифровых технологий
Кафедра КБ-3 «Разработка программных решений и системное программирование»
Дисциплина «Алгоритмы и структуры данных»

Проверочная работа по теме «Анализ сложности нерекурсивных функций»

Вариант № 7

Группа: БСБО-16-23	№ студ. билета 23Б0107	Студент (ФИО): Крашенинников Матвей Вячеславович

Пусть фрагмент программы содержит две функции, причем одна функция содержит вызов (вызовы) другой. Назовем первую функцию – вызывающей (которая для определенности запускается из main()), а вторую – вызываемой.

На основе пооператорного анализа исходного кода программы определите функции роста трудоемкости *f*(*N*) каждой функции (и вызываемой, и вызывающей) отдельно и их точные асимптотические оценки Θ(·) (*или* O(·)).

Кроме того, по возможности определите возвращаемое значение вызываемой функции.

```
#include <iostream>

int D(int n, long& s) { // 2
   int i, j, x = 10; // + 1
   for (i = 1; i <= 2 * n; i += 2, x += 4) { // + 1 + 2 + ∑[1; n](2 + 1 + 1 + ...
      for (j = 1; j <= n; j += 1, x++) { // + 1 + 1 + ∑[1; n](1 + 1 + 1 + ...
         s = s + i * j; // + 1 + 1 + 1))
      }
   }

   while (j <= 3 * n) { // 2 + ∑[j'=1; log2(3n/(n+1))](2 + ...
      j = j + j; // j = j * 2 | + 2
      s = s + j; // + 2)
   }

   return x; // + 1
}

void DD(void) {
   int t, a = -1, b = -1; // + 2
   long S = 0; // + 1
   int n;  std::cout << "n = "; std::cin >> n; // + 2

   for (t = 1; t <= n; t++) { // + 1 + 1 + ∑[1; n](1 + 1 + ...
      if (D(t, S) > 20) { // + 2 + 1
         a = D(t, S); // + 1 + 2
      } else {
         b = D(n, S) + D(20, S); // + 1 + 2 + 1 + 2
      }
   }

   std::cout << "\na = " << a << "   b = " << b; // + 4
   std::cout << "   S = " << S << "\n"; // + 3
}

int main() {
   DD();
}
```

Определим трудоёмкость выполнения и её асимптотическую оценку вызываемой функции D:

$F(n) = 1+1+2+\sum_{i=1}^{n}(2+1+1+1+1+\sum_{j=1}^{n}(1+1+1+1+1+1))+2+\sum_{j'=1}^{\log_2(\frac{3*n}{n+1})}(2+2+2)+1=$

$= 7+\sum_{i=1}^{n}(6+\sum_{j=1}^{n}(6))+\sum_{j'=1}^{\log_2(\frac{3*n}{n+1})}(6)=$

$=7+\sum_{i=1}^{n}(6+6*\sum_{j=1}^{n}(1))+6*\sum_{j'=1}^{\log_2(\frac{3*n}{n+1})}(1)=$

$=7+\sum_{i=1}^{n}(6+6n)+6\log_2(\frac{3*n}{n+1})=$

$=7+\sum_{i=1}^{n}(6)+\sum_{i=1}^{n}(6n)+6\log_2(\frac{3*n}{n+1})=$

$=7+6n+6n^2+6\log_2(\frac{3*n}{n+1})$

Таким образом, $O(f(n)) = n^2$.

Заметим, что в функции DD() есть ветвление “if-else”. Рассмотрим D(t, S): очевидно, что x стремится к бесконечности, тогда возьмём минимально возможное t = 1 для адекватной работы программы. При этом значении D(t, S) = 15, значит, попадаем в else. Если t = 2, имеем: D(t, S) = 22. Получается, что при t > 1 всегда исполняться будет “if”.

Посчитаем ветку “else” – с ней работать будем «вне цикла», а с “if” в цикле, начиная с 2.

<else>$=2+1+D(1, 0)+1+2+1+2+D(n, S)+D(20, S)+1=10+19+6\log_2(1{,}5)+7+6n+6n^2+6\log_2(\frac{3*n}{n+1})+2527+6\log_2(\frac{20}{7})=2563+6*(n+n^2+\log_2(\frac{3*n}{n+1})+\log_2(1{,}5)+\log_2(\frac{20}{7}))$

1) $D(1, 0)=7+6*1+6*1*1+6\log_2(\frac{3*1}{1+1})=7+6+6+6\log_2(1{,}5)=19+6\log_2(1{,}5)$

2) $D(n, S)=7+6n+6n^2+6\log_2(\frac{3*n}{n+1})$

3) $D(20, S)=7+6*20+6*20*20+6\log_2(\frac{3*20}{20+1})=2527+6\log_2(\frac{20}{7})$

Исходя из этого:

<table>
<tr>
<td></td>
<td>
F(n) = $\sum_{t=2}^{n}(1+1+2+1+1+2+2*D(t,S))$+<else>=$\sum_{t=2}^{n}(8+2*D(t,S))$+<else>=
=$\sum_{t=2}^{n}(8)+\sum_{t=2}^{n}(2*D(t,S))$+<else>=
=$8*n-8+2563+6*(n+n^2+\log_2(\frac{3*n}{n+1})+\log_2(1,5)+\log_2(\frac{20}{7}))+\sum_{t=2}^{n}\left(2*(7+6t+6t^2+6\log_2(\frac{3*t}{t+1}))\right)$=$8n-8+2563+6*(n+n^2+\log_2(\frac{3*n}{n+1})+\log_2(1,5)+\log_2(\frac{20}{7}))+\sum_{t=2}^{n}\left(2*(7+6t+6t^2+6\log_2(\frac{3*t}{t+1}))\right)$=$8n+2555+6*(n+n^2+\log_2(\frac{3*n}{n+1})+\log_2(1,5)+\log_2(\frac{20}{7}))+\sum_{t=2}^{n}\left(14+12t+12t^2+12\log_2(\frac{3*t}{t+1})\right)$*=$8n+2555+6n+6n^2+6\log_2(\frac{3*n}{n+1})+6\log_2(1,5)+6\log_2(\frac{20}{7})+22n-38+12n^2+4n^3+12n\log_2(3)-12\log_2(3)$=
=$36n+2523+18n^2+6\log_2(n)-6\log_2(n+1)+6\log_2(5/7)+4n^3+12n\log_2(3)$
*=$14\sum_{t=2}^{n}(1)+12\sum_{t=2}^{n}(t)+12\sum_{t=2}^{n}\left(t^2\right)+12\sum_{t=2}^{n}\left(\log_2(\frac{3*t}{t+1})\right)$=$14n-14+12(n(n+1)/2-1)+$
$+12(n(n+1)(2n+1)/6-1)+12\log_2(3)(n-1)$=
=$22n-38+12n^2+4n^3+12n\log_2(3)-12\log_2(3)$
Оценка на основе того, что $\log_2(\frac{3t}{t+1}) \sim \log_2(3)$

Возвращаемое значение вызываемой функции участвует в небольшом участке кода:
<pre>
int i, j, x = 10;
   for (i = 1; i <= 2 * n; i += 2, x += 4) {
      for (j = 1; j <= n; j += 1, x++)
</pre>
Заметим: инициализация x = 10.
Внешний цикл выполняется n раз, поскольку его диапазон [1;2*n], а итератор увеличивается на 2, X же – на 4. Значит, к общей формуле добавляем 4*n. Внутренний цикл выполняется n раз (как и внешний), поэтому оставшийся член формулы – это n^2.
</td>
</tr>
<tr>
<td>
Ответы:
1. Возвращаемое значение вызываемой функции (в общем виде)
2. Трудоемкость выполнения и её асимптотическая оценка вызываемой функции
3. Трудоемкость выполнения и её асимптотическая оценка вызывающей функции (всей целиком)
</td>
<td>
1. D(n, s) = $n^2 + 4n + 10$
2. F(n)=$7+6n+6n^2+6\log_2(\frac{3*n}{n+1})$; O(f(n)) = n^2
3. F(n)=$36n+2523+18n^2+6\log_2(n)-6\log_2(n+1)+6\log_2(5/7)+4n^3+12n\log_2(3)$; O(f(n)) = n^3
</td>
</tr>
</table>