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 $\Theta(\cdot)$ (*или* $O(\cdot)$).

Кроме того, по возможности определите возвращаемое значение вызываемой функции.

```cpp
#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:

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

$= 5+\sum_{i=1}^{n}(6+\sum_{j=1}^{n}(6))+\sum_{j'=1}^{\log_2(\frac{3*n}{n+1})}(6)=$

$=5+\sum_{i=1}^{n}(6+6*\sum_{j=1}^{n}(1))+6*\sum_{j'=1}^{\log_2(\frac{3*n}{n+1})}(1)=$

$=5+\sum_{i=1}^{n}(6+6n)+6\log_2(\frac{3*n}{n+1})=$

$=5+\sum_{i=1}^{n}(6)+\sum_{i=1}^{n}(6n)+6\log_2(\frac{3*n}{n+1})=$

$=5+6n+6n^2+6\log_2(3)+6\log_2(n)-6\log_2(n+1)$

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 n = 1 для адекватной работы программы. При этом значении D(t, S) = 15, значит, попадаем в else. Если n = 2, имеем: D(t, S) = 22. Получается, что при n > 1 всегда исполняться будет if.

Исходя из анализа, выберем наиболее частый сценарий:

$F(n) = 2+1+2+1+1+\sum_{t=1}^{n}(1+1+2+1+1+2+5+6n+6n^2+6\log_2(3)+6\log_2(n)-6\log_2(n+1))+4+3=$

$=14+13n+6n^2+6n^3+6n\log_2(3)+6n\log_2(n)-6n\log_2(n+1)$

Получается, что $O(f(n)) = n^3$

Возвращаемое значение вызываемой функции участвует в небольшом участке кода:

```cpp
int i, j, x = 10;
   for (i = 1; i <= 2 * n; i += 2, x += 4) {
      for (j = 1; j <= n; j += 1, x++)
```

	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.
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)	1. $D(n, s) = n^2 + 4n + 10$ 2. $F(n)=5+6n+6n^2+6\log_2(3)+6\log_2(n)-6\log_2(n+1)$; $O(f(n)) = n^2$ 3. $F(n)=14+13n+6n^2+6n^3+6n\log_2(3)+6n\log_2(n)-6n\log_2(n+1)$; $O(f(n)) = n^3$