Филиппова Анастасия

Домашнее задание 3

(1) Можно брать любые возможные 2 числа.Тогда поступим следующим образом: возьмем любую пару чисел с помощью алгоритма Евклида придем к НОД этой пары чисел.Вместо этой пары чисел запишем их НОД.С остальными числами поступим аналогично.В итоге придем к тому, что на доске будет написаны одинаковые числа,равные наибольшему общему делителю всех чисел, написанных на доске в начале.
Ответ: НОД всех чисел.

(2) Разложим числа x и y на простые множители: $x = p_1^{k_1}...p_n^{k_n}, b = p_1^{t_1}...p_n^{t_n}$, тогда НОД$(x, y) = p_1^{min(t_1,k_1)}...p_n^{min(t_n,k_n)}$, НОК$(x, y) = p_1^{max(t_1,k_1)}...p_n^{max(t_n,k_n)}$, тогда $x \cdot y =$НОК·НОД.
Находим НОД(x, y) используя расширенный алгоритм Евклида:
Псевдо - код:

```
Euclid (x, y)  if (y == 0) then
 |  return x; return Euclid(y, x mod y)
end
```

Сложность алгоритма в худшем случае $O(n^3)$: $2n$ рекурсивных вызовов и при каждом происходит деление, которое требует $O(n^2)$ времени. И того $O(n^3)$.
Далее премножаем числа a, b по следующему алгоритму:
Псевдо - код:

```
Multiply(x, y)  if (y == 0) then
 |  return 0; z = Multiply(x; y/2);
end
if (y mod 2 == 0) then
 |  return 2z;
end
else
 |  return x + 2z
end
```

Сложность: $O(n^2)$ ($O(n)$ битовых операций n раз).
Далее делим произведение на НОД.Сложность $O(n^2)$. Получили алгоритм нахождения НОК 2-х чисел с общем временем работы $O(n^3)$.

(3)

(4) (a) $T(n) = 36T(\frac{n}{6}) + n^2$
$a = 36, b = 6, n^{\log_b a} = n^2$
$f(n) = \Theta(n^2) \Rightarrow$ по 2-ому пункту мастер-теормемы $T(n) = \Theta(n^2 \cdot \log n)$

(b) $T(n) = 3T(\frac{n}{3} + n^2)$
$a = 3, b = 3, n^{\log_b a} = 1$
$f(n) = \Omega(n^{(\log_b a)+\varepsilon}) = O(n)$, где $\varepsilon = 1$ тогда по 3-ему пункту мастер-теоремы $T(n) = \Theta(n^2)$.

(c) $T(n) = 4T(\frac{n}{2}) + \frac{n}{\log n}$
$a = 4, b = 2, n^{(\log_b a)} = n^2$
$f(n) = O(n^{2-\varepsilon})$, где $\varepsilon = 1$, по 1-ому пункту основной теоремы о рекурсиях получаем: $T(n) = \Theta(n^2)$.

(5) $T(n) = nT(n/2) + O(n) = n(T(n/2) + O(1)) = n(n/2(T(n/4)) + O(n/2) + O(1)) = n(n/2(T(n/4) + O(1)) + O(1)$
$O(1)$ всегда меньше чем какая-то константа, тогда при больших n можем ей пренебречь, в то время как для мальньких n функция $T(n)$ ограничена снизу какой-то константой(так как $T(1)$ выполняется за константное время), следовательно можно пренебрчь $O(n)$,на k - ом шаге: $\frac{n^{k+1}}{2^{\frac{k(k+1)}{2}}}$, высота дерева рекурсии $k = \log_2 n$, тогда, так как на последнем уровне $T(1)$ - константное время, получаем $\frac{n^{\log_2 n+1}}{2^{\frac{\log_2 n(\log_2 n+1)}{2}}} = \frac{n^{\log_2 n}}{\sqrt{n^{\log_2 n} n}} = n^{\log_2 \sqrt{2n}}$
Ответ: $T(n) = \Theta(n^{\log_2 \sqrt{2n}})$

(6) (a) $T(n) = T(\alpha n) + T((1 - \alpha)n) + \Theta(n)$
Округления в каждом рекурсионном вызове вниз, поэтому можем им пренебречь.
$T(n) = T(\alpha^2 n) + 2T(\alpha n + ((1 - \alpha)n)) + T((1 - \alpha)^2 n) + 2\Theta(n)$, тогда на k - ом шаге: $T(n) = \sum_{i=0}^{k} C_k^i T(\alpha^{k-i}(1 - \alpha)^i) + kCn$
В зависимости от значения параметра α правая или левая ветвь дойдут до $T(1)$ быстрее (первой или последней).Пусть $x = max(\alpha, (1 - \alpha))$, тогда оценим функцию снизу и сверху 2-мя деревьями: $\log_{1/(1-x)} n \cdot n <= T(n) <= \log_{1/x} n \cdot n \Rightarrow T(n) = \Theta(n \cdot \log n)$

(b) Округления везде вниз, поэтому ими можем пренебречь.
$T(n) = T(n/2) + 2T(n/4) + \Theta(n)$
Левая ветвь построенного дерева рекурсии имеет вид:$T(n) \to T(n/2) \to T(n/4) \to ... \to T(n/2^k) \to ... \to T(1)$
Правая:
$T(n) \to T(n/4) \to T(n/16) \to ... \to T(n/4^k) \to ... \to T(1)$
Правая ветка дойдет до $T(1)$ первой, а левая последней.Высота всего дерева $\log_2 n$, высота дерева, обрезанного до конца правой ветви $\log_4 n$. Получаем верхнюю и нижнюю оценки:
$c_1 n \log_2 n <= T(n) <= c_2 \log_2 n \Rightarrow T(n) = \Theta(n \log n)$

(c) $T(n) = 27T(n/3) + \frac{n^3}{\log^2 n} = 27^2 T(n/9) + \frac{n^3}{\log^2 \frac{n}{3}} + \frac{n^3}{\log^2 n}$, на k - ом шаге:$T(n) = 27^k T(n/3^k) + \sum 27^{k-1} \frac{\frac{n}{3^{k-1}}}{3 \log^2(\frac{n}{3^{k-1}})}$
Высота дерева $\log_3 n$.Оценим каждое слагаемое суммы по отдельности:
$T(n) = 27^{\log_3 n} T(1) = n^3$
$\sum\limits_{k=0}^{\log_3 n - 1} \frac{n^3}{\log^2 \frac{n}{3^k}}$, "выкинем"константу $\log 3 : n^3 \sum\limits_{i=0}^{(\log_3 n - 1)} \frac{1}{(\log n) - i}$
$0 <= \sum\limits_{i=0}^{(\log_3 n - 1)} \frac{1}{((\log n) - i)^2} < \int_0^{\log n - 1} \frac{1}{((\log n) - i)^2} di = \int_0^{\log n - 1} - \frac{1}{((\log n) - i)^2} d(\log n - i) = 2 - \frac{2}{\log n} <= 2 \Rightarrow \frac{n^3}{\log^2 \frac{n}{3^k}} = \Theta(1) \Rightarrow T(n) = \Theta(n^3)$.

(7) (a) Нужно массив размером n заполнить $(i!)^{-1}$ $mod\ p$.Сначала вычисляем $n!$ $mod\ p$.Далее с помощью расширенного алгоритма Евклида находим $(n!)^{-1}$ $mod\ p$.Далее, зная, что $k!(k!)^{-1} \underset{mod\ p}{\equiv} 1 \underset{mod\ p}{\equiv} ((k-1)!)((k-1)!)^{-1} \Rightarrow ((k-1)!)^{-1} \underset{mod\ p}{\equiv} k(k!)^{-1}$ запомняем массив.
Докажем корректность данного алгоритма:
Корректность очевидна, так как i пробегает от 1 n, и для каждого i выполняется правило, написанное выше.
Сложность:
Сначала вычисляем $n!$ $mod\ p$ за $O(n)$ операций (так как не зависит от длины входа числа n, так как длина входа определенная(константа)).Далее с помощью расширенного алгоритма Евклида находим $(n!)^{-1}$ $mod\ p$.Сложность данной операции зависит от длины входа числа p , то есть $\log p$. Считая, что $\log p = O(n)$, находим $(n!)^{-1}$ $mod\ p$ за O(n). Умножение стоит $O(1) \Rightarrow$данный алгоритм линейный.

(7)(с семинара)
$ab = \frac{(a+b)^2 + a^2 + b^2}{2}$
Пусть длина наибольшего из входов n.Докажем, что данный алгоритм вычисления произведения работает за линейное время.Действительно: сложение стоит $O(n)$, возведение в квадрат по условию стоит $O(n)$, деление на 2 тоже $O(n)$.Получаем 3 операции за $O(n) \Rightarrow$алгоритм работает за время $O(n)$.