Примерни решения на изпита по ДАА на Информатика и КН 2 поток, проведен на 24.06.2018 г.

Зад. 1 Разгледайте следния алгоритъм.

```
Alg1(n: цяло число, по-голямо или равно на 8)
 1  i ← 8, p ← 1, q ← 1
 2  while i < n do
 3      i++
 4      if p ≥ 1
 5          p--
 6          q ← q + 2
 7      else
 8          p ← p + 2
 9          q ← q − 3
10  return (p, q)
```

Докажете, че алгоритъмът връща такива p и q, че $p \geq 0$, $q \geq 0$ и $n = 5p + 3q$.

Решение: По условие, $n \geq 8$. Ако $n = 8$, **while**-цикълът (редове 2–9) не се изпълнява, защото i получава стойност 8 на ред 2. Изпълнението отива на ред 10, където алгоритъмът връща $(1, 1)$, тъй като на ред 1 има $p \leftarrow 1$ и $q \leftarrow 1$. Вярно е, че алгоритъмът връща такива p и q, че $p \geq 0$, $q \geq 0$ и $n = 5p + 3q$.

Остава да покажем, че твърдението е вярно при $n > 8$. Следното е инвариант за **while**-цикълът (редове 2–9):

> При всяко достигане на ред 2 е вярно, че $i = 5p + 3q$, $i \geq 8$, $p \geq 0$ и $q \geq 0$.

База: При първото достигане на ред 2, i е 8, а p и q са единици заради присвояванията на ред 1. Инвариантът е в сила.

Поддръжка: Да допуснем, че инвариантът е в сила за някое достигане на ред 2, което не е последното. Нека i', p' и q' са стойностите съответно на i, p и q в този момент. На ред 3, i се инкрементира, така че $i = i' + 1$.

- Да допуснем, че $p' \geq 1$. Тогава условието на ред 4 е истина и след изпълнението на редове 5 и 6 имаме $p = p' - 1$ и $q = q' + 2$. Съгласно индуктивното предположение, $i' = 5p' + 3q'$. Но

  $$\begin{aligned} & i' = 5p' + 3q' \ \leftrightarrow \\ & i' + 1 = 5p' + 3q' + 1 \ \leftrightarrow \\ & i' + 1 = 5p' + 3q' - 5 + 6 \ \leftrightarrow \\ & i' + 1 = 5(p' - 1) + 3(q' + 2) \ \leftrightarrow \\ & i = 5p + 3q \end{aligned}$$

  И така, при следващото достигане на ред 2 отново е вярно, че $i = 5p + 3q$. Нещо повече, $p \geq 0$, защото $p' \geq 1$, а $q \geq 0$, защото $q' \geq 0$ по допускане, а $q = q' + 2$. Това, че $i \geq 8$, следва веднага от допускането $i' \geq 8$ и факта, че $i = i' + 1$. Виждаме, че инвариантът остава в сила.

- Да допуснем, че $p' < 1$. Тъй като по допускане $p' \geq 0$, единствено възможно е $p' = 0$.

  Ще покажем, че $q' \geq 3$. Наистина, ако допуснем, че $q' \leq 2$, достигаме до противоречие, защото в текущите допускания, $i' = 5p' + 3q'$ и $p' = 0$, което влече $i' = 3q'$; това е невъзможно, ако $i' \geq 8$ и $q' \leq 2$.

И така, $q' \geq 3$. След изпълнението на редове 8 и 9 имаме $p = p' + 2$ и $q = q' - 3$. Съгласно индуктивното предположение, $i' = 5p' + 3q'$. Но

$$
\begin{aligned}
& i' = 5p' + 3q' \ \leftrightarrow \\
& i' + 1 = 5p' + 3q' + 1 \ \leftrightarrow \\
& i' + 1 = 5p' + 3q' - 9 + 10 \ \leftrightarrow \\
& i' + 1 = 5(p' + 2) + 3(q' - 3) \ \leftrightarrow \\
& i = 5p + 3q
\end{aligned}
$$

И така, при следващото достигане на ред 2 отново е вярно, че $i = 5p + 3q$. Нещо повече, $p \geq 0$, защото $p' = 0$, а $p = p' + 2$; освен това, $q \geq 0$, защото $q' \geq 3$ и $q = q' - 3$. Това, че $i \geq 8$, следва веднага от допускането $i' \geq 8$ и факта, че $i = i' + 1$. Виждаме, че инвариантът остава в сила.

Терминация: При последното достигане на ред 2 очевидно $i = n$. Заместваики в инварианта, получаваме "$n = 5p + 3q$, $p \geq 0$ и $q \geq 0$". Тогава за наредената двойка (p, q), която алгоритъмът връща на ред 10, е изпълнено желаното твърдение.

Зад. 2 Подредете по асимптотично нарастване следните 12 функции:

$$f_1(n) = n^2 \qquad f_2(n) = n\binom{n}{2} \qquad f_3(n) = \sum_{i=1}^{n} \binom{n}{i}^2 \qquad f_4(n) = \sum_{i=1}^{\infty} \frac{n^2}{2^i}$$

$$f_5(n) = \sqrt[n]{n!} \qquad f_6(n) = \lg\lg\lg n \qquad f_7(n) = 2^{2^n} \qquad f_8(n) = 2^{2^{n+2}}$$

$$f_9(n) = 2^{n^2} \qquad f_{10}(n) = \lg\lg\lg\binom{n}{2} \qquad f_{11}(n) = 2^{n!} \qquad f_{12}(n) = \sum_{k=1}^{n^2} \frac{n^2}{k}$$

Напишете окончателната наредба в явен вид.

Решение Да сравним $f_6(n)$ с $f_{10}(n)$. Тъй като $\binom{n}{2} \asymp n^2$, в сила е

$$f_{10}(n) \asymp \lg\lg\lg n^2 = \lg\lg(2\lg n) \asymp \lg\lg\lg n$$

Тогава

$$f_{10}(n) \asymp f_6(n)$$

Да сравним $f_6(n)$ с $f_5(n)$. Ще опростим $f_5(n)$, използвайки апроскимацията на Stirling:

$$\sqrt[n]{n!} \asymp \sqrt[n]{\sqrt{n}\frac{n^n}{e^n}} = \sqrt[2n]{n}\frac{n}{e} \asymp n\sqrt[2n]{n}$$

Очевидно $\lg\lg\lg n \prec n\sqrt[2n]{n}$, така че $f_6(n) \prec f_5(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n)$$

Да сравним $f_5(n)$ с $f_1(n)$. Очевидно е, че $n\sqrt[2n]{n} \prec n^2$, така че $f_5(n) \prec f_1(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n)$$

Да сравним $f_1(n)$ с $f_4(n)$. Очевидно $f_4(n) = n^2 \sum_{i=1}^{\infty} \frac{1}{2^i}$, понеже n не зависи от i. Известно е, че геометричният ред $\sum_{i=1}^{\infty} \frac{1}{2^i}$ е сходящ, поради което $f_4(n) \asymp n^2$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n)$$

Да сравним $f_1(n)$ с $f_{12}(n)$. Очевидно $f_{12}(n) = n^2 \sum_{k=1}^{n^2} \frac{1}{k}$. От лекции знаем, че $\sum_{k=1}^{m} \frac{1}{k} \asymp \lg m$, откъдето $f_{12}(n) \asymp n^2 \lg n^2 \asymp n^2 \lg n$. Тогава $f_1(n) \prec f_{12}(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n)$$

Да сравним $f_{12}(n)$ с $f_2(n)$. Тъй като $\binom{n}{2} \asymp n^2$, в сила е $f_2(n) \asymp n^3$. Тогава $f_{12}(n) \prec f_2(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n)$$

Да сравним $f_2(n)$ с $f_3(n)$. От курса Дискретни Структури знаем, че $\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n}$. А на лекция по ДАА сме показали, че $\binom{2n}{n} \asymp \frac{4^n}{\sqrt{n}}$. Тогава $f_3(n) \asymp \frac{4^n}{\sqrt{n}}$, така че $f_2(n) \prec f_3(n)$.

Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n) \prec f_3(n)$$

Да сравним $f_3(n)$ с $f_9(n)$. Както видяхме, $f_3(n) \asymp \frac{4^n}{\sqrt{n}}$, а $f_9(n) = 2^{n^2}$ по условие. Ако логаритмуваме двете страни на двете асимптотични еквивалентности, получаваме

$$\lg f_3(n) \asymp n$$
$$\lg f_9(n) \asymp n^2$$

Тъй като $n \prec n^2$, за първообразите директно имаме $f_3(n) \prec f_9(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n) \prec f_3(n) \prec f_9(n)$$

Да сравним $f_9(n)$ с $f_7(n)$. Логаритмувайки, получаваме

$$\lg f_9(n) \asymp n^2$$
$$\lg f_7(n) \asymp 2^n$$

Тъй като експоненциалната 2^n расте по-бързо, в асимптотичния смисъл, от полиномиалната n^2, за първообразите е в сила $f_9(n) \prec f_7(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n) \prec f_3(n) \prec f_9(n) \prec f_7(n)$$

Да сравним $f_7(n)$ с $f_8(n)$. Забелязваме, че

$$\left(2^{2^n}\right)^4 = 2^{2^2 \cdot 2^n} = 2^{2^{n+2}}$$

Тогава $f_8(n) = (f_7(n))^4$. Тъй като това са неограничено растящи функции, вярно е, че $f_7(n) \prec f_8(n)$. Дотук имаме

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n) \prec f_3(n) \prec f_9(n) \prec f_7(n) \prec f_8(n)$$

Да сравним $f_8(n)$ с $f_{11}(n)$. Логаритмувайки, получаваме

$$\lg f_8(n) \asymp 2^{n+2} = 4 \cdot 2^n \asymp 2^n$$
$$\lg f_{11}(n) \asymp n! \asymp \sqrt{n}\frac{n^n}{e^n}$$

Ако логаритмуваме още веднъж, получаваме

$$\lg \lg f_8(n) \asymp n$$
$$\lg \lg f_{11}(n) \asymp n \lg n$$

Тъй като $n \lg n$ расте по-бързо, в асимптотичния смисъл, от n, в сила е $f_8(n) \prec f_{11}(n)$. Окончателната наредба е

$$f_{10}(n) \asymp f_6(n) \prec f_5(n) \prec f_1(n) \asymp f_4(n) \prec f_{12}(n) \prec f_2(n) \prec f_3(n) \prec f_9(n) \prec f_7(n) \prec f_8(n) \prec f_{11}(n)$$

Зад. 3 Докажете, че за да бъдат намерени максимума и минимума на n числа чрез, и само чрез, директни сравнения, са необходими поне $\lceil \frac{3n}{2} \rceil - 2$ сравнения.

Бонус 10 точки: Ако освен това напишете алгоритъм, използващ точно толкова сравнения, ще получите бонус от 10 точки. Бонус може да има само при наличие на смислен, може би непълен, аргумент за долната граница.

Решение: Това е разглеждано в час.

Зад. 4 Професор Дълбоков предлага следния алгоритъм за намиране на МПД, изграден по схемата **Разделяй-и-Владей**. Нека $G = (V, E)$ е неориентиран свързан тегловен граф.

- Ако $|V| = 1$, алгоритъмът връща единствения връх от V.
- Ако $|V| > 1$, алгоритъмът разбива по произволен начин V на непразни V_1 и V_2, такива че $|V_1|-|V_2| \in \{-1, 0, +1\}$, и вика себе си рекурсивно върху подграфите, индуцирани от V_1 и V_2, като получава съответно МПД-та T_1 и T_2. След това добавя към T_1 и T_2 произволно най-леко ребро, прекосяващо среза $\{V_1, V_2\}$, и връща полученото дърво T.

Ако алгоритъмът е коректен, обосновете в общи линии коректността му. Ако алгоритъмът не е коректен, докажете, че не е коректен.

Решение: Алгоритъмът не е коректен. Ето контрапример. Нека

$$G = (\{a, b, c, d\}, \{(a, b), (b, c), (c, d), (b, d), (a, d)\})$$

и тегловната функция w е следната: $w(a, b) = 10$, $w(b, c) = 1$, $w(c, d) = 10$, $w(b, d) = 1$, $w(a, d) = 1$.

Нека първото разбиване е $\{\{a, b\}, \{c, d\}\}$.

- Викайки върху подграфа, индуциран от $\{a, b\}$, алгоритъмът разбива на $\{\{a\}, \{b\}\}$, като двете викания веднага връщат, а към ПД-то се добавя реброто (a, b) с тегло 10.
- Напълно аналогично се добавя реброто (c, d), също с тегло 10.

Най-леко ребро, прекосяващо среза $\{\{a, b\}, \{c, d\}\}$, е с тегло 1. Алгоритъмът го добавя и него и ПД-то се оказва с тегло 21.

В действителност МПД-то е с тегло 3.

Зад. 5 Дадени са n къщи $h_1, h_2, \ldots, h_n$, които са една до друга в редица в точно този ред. Всяка къща трябва да бъде боядисана в един от цветовете червен, зелен или син. Никои две съседни къщи не трябва да са в един и същи цвят. За всяка къща h_i са известни

- цената $c_r[i]$ за боядисване на h_i в червен цвят,
- цената $c_g[i]$ за боядисване на h_i в зелен цвят,
- цената $c_b[i]$ за боядисване на h_i в син цвят.

Предложете колкото е възможно по-ефикасен алгоритъм за изчисляване на минималната цена за боядисване на всички къщи. Достатъчно е Вашият алгоритъм да намира само цената, а не и коя къща в какъв цвят да бъде.

Съвсем накратко обосновете коректността и сложността по време и памет на Вашия алгоритъм.

Решение: Можем да разсъждаваме така. Едната от крайните къщи, да кажем h_n, трябва да бъде боядисана или в червено, или в зелено, или в синьо.

- Ако h_n бъде боядисана в червено, минималната цена за всички къщи е $c_r[n]$ плюс минималната цена за боядисване на $h_1, \ldots, h_{n-1}$, като h_{n-1} може да е зелена или синя.
- Ако h_n бъде боядисана в зелено, минималната цена за всички къщи е $c_g[n]$ плюс минималната цена за боядисване на $h_1, \ldots, h_{n-1}$, като h_{n-1} може да е червена или синя.
- Ако h_n бъде боядисана в синьо, минималната цена за всички къщи е $c_b[n]$ плюс минималната цена за боядисване на $h_1, \ldots, h_{n-1}$, като h_{n-1} може да е червена или зелена.

Нека $\mathrm{opt}_r[i]$ е минималната цена за боядисване на къщи $h_1, \ldots, h_i$ по такъв начин, че h_i е червена. Нека $\mathrm{opt}_g[i]$ е минималната цена за боядисване на къщи $h_1, \ldots, h_i$ по такъв начин, че h_i е зелена. Нека $\mathrm{opt}_b[i]$ е минималната цена за боядисване на къщи $h_1, \ldots, h_i$ по такъв начин, че h_i е синя.

Рекурсивната декомпозиция е следната.

$$\mathrm{opt}_r[1] = c_r[1]$$
$$\mathrm{opt}_g[1] = c_g[1]$$
$$\mathrm{opt}_b[1] = c_b[1]$$
$$\mathrm{opt}_r[i] = c_r[i] + \min\{\mathrm{opt}_g[i-1], \mathrm{opt}_b[i-1]\} \text{ за } i \geq 2$$
$$\mathrm{opt}_g[i] = c_g[i] + \min\{\mathrm{opt}_r[i-1], \mathrm{opt}_b[i-1]\} \text{ за } i \geq 2$$
$$\mathrm{opt}_b[i] = c_b[i] + \min\{\mathrm{opt}_r[i-1], \mathrm{opt}_g[i-1]\} \text{ за } i \geq 2$$

Търсеният отговор е $\min\{\mathrm{opt}_r[n], \mathrm{opt}_g[n], \mathrm{opt}_b[n]\}$.

Превръщането на тази рекурсивна декомпозиция в алгоритъм по схемата **Динамично Програмиране** е тривиално.

```
Боядисай евтино(c_r[1,...,n], c_g[1,...,n], c_b[1,...,n])
1  opt_r[1] ← c_r[1]
2  opt_g[1] ← c_g[1]
3  opt_r[1] ← c_b[1]
4  for i ← 2 to n
5      opt_r[i] ← c_r[i] + min(opt_g[i−1], opt_b[i−1])
6      opt_g[i] ← c_g[i] + min(opt_r[i−1], opt_b[i−1])
7      opt_b[i] ← c_b[i] + min(opt_r[i−1], opt_g[i−1])
8  return min(opt_r[n], opt_g[n], opt_b[n])
```

Коректността е очевидна предвид коректността на рекурсивната декомпозиция. Сложността и по време, и по памет е $\Theta(n)$.

Зад. 6 Разгледайте тази задача: даден е неориентиран свързан граф G и естествено число k и се търси такова покриващо дърво T на G, че максималната степен на връх в T е не по-голяма от k. Какво можете да кажете за сложността на задачата?

Решение: Задачата е **NP**-пълна, ако е във вариант за разпознаване, и е разглеждана на лекция. Принадлежността към **NP** е очевидна. Има тривиална редукция от Хамилтонов път: при $k = 2$, покриващото дърво е Хамилтонов път.