Примерни решение на поправителния изпит по ДАА на 26 август 2023 г.

Зад. 1 Ако $D = (V_D, E_D)$ е ориентиран граф, *свръхсифон* в D е всеки сифон u, такъв че за всеки $v \in V_D \setminus \{u\}$ е вярно, че $(v, u) \in E_D$.

Даден е ориентиран граф $G = (\{1, 2, \ldots, n\}, E)$, представен с **матрица на съседство**. Предложете алгоритъм със сложност $O(n)$, такъв че

- ако в G има свръхсифон, връща i, където i е някой свръхсифон в G,
- ако в G няма свръхсифон, връща -1.

Обосновете много прецизно и подробно коректността на Вашия алгоритъм. Очаква се подробен псевдокод и доказателство за коректност чрез инвариант на цикъл, ако алгоритъмът Ви е рекурсивен, или чрез индукция по големината на входа, ако алгоритъмът Ви е итеративен.

Ако използвате прости, очевидни помощни алгоритми като процедури, допустимо е да не доказвате тяхната коректност, а да я вземете за очевидна. Но коректността на основния алгоритъм трябва да се докаже безукорно.

Решение: На прост български, свръхсифон е връх, в който влиза по едно ребро от всеки друг връх, но не излиза нито едно ребро. Тъй като от всеки друг връх трябва да има ребро към свръхсифона, очевидно е, че ако има свръхсифон, той е единствен. На английски приетият термин е *universal sink*.

Да си представим матрицата на съседство на ориентиран граф, в който връх i е свръхсифон: ред i на матрицата съдържа само нули, а колона i на матрицата съдържа само единици, с изключение на клетка $[i, i]$, която съдържа нула.

Във време $O(n)$ можем да проверим дали връх i е свръхсифон, за дадено i. Но как да изчислим във време $O(n)$ дали има свръхсифон? Очевидният, наивен подход е да проверим за $i \in \{1, \ldots, n\}$ дали i е свръхсифон, преглеждайки клетките на ред i и колона i. Това води до $\Omega(n^2)$ алгоритъм. Трябва да подходим по-ефикасно.

Нека матрицата на съседство е $M[1\,..\,n, 1\,..\,n]$. Тъй като графът няма примки, главният диагонал съдържа само нули. Да мислим за произволна клетка $M[i, j]$, такава че $i > j$. Ключовото наблюдение е, че ако $M[i, j] = 0$, то връх j не може да е свръхсифон, понеже няма ребро от i до него.

Това наблюдение подсказва идея за алгоритъм със сложност $O(n)$. Инициализираме $i \leftarrow 1$ и $j \leftarrow 1$ и кандидатът за свръхсифон става i.

- Ако текущата клетка $M[i, j]$ съдържа нула и $j < n$, правим "ход надясно" $j{+}{+}$ и кандидатът за свръхсифон се запазва.
- Ако текущата клетка съдържа единица, новата текуща клетка става $M[j, j]$, а кандидатът за свръхсифон става j.

Когато j стане n, проверяваме дали кандидадът за свръхсифон наистина е свръхсифон: дали ред i се състои само от нули, а колона i (без клетка $M[i, i]$), само от единици. Ето псевдокод.

```
UNIVERSAL SINK(M[1 .. n, 1 .. n]: матрица на съседство)
 1  (* M[i,i] = 0 за 1 ≤ i ≤ n *)
 2  i ← 1, j ← 1
 3  while TRUE do
 4      while j ≤ n and M[i,j] = 0 do
 5          j++
 6      if j = n + 1
 7          res ← TEST(M, i)
 8          if res = TRUE
 9              return i
10          else
11              return −1
12      else
13          i ← j
```

Забележете, че алгоритъмът ползва конвенцията на езика C, според която в булево условие-конюнкция, оценяването е отляво надясно и, ако някой операнд се окаже със стойност FALSE, останалите операнди не се оценяват: на ред 4, ако се окаже, че $j > n$, не се прави проверка дали $M[i,j]$ е нула, и това е напълно резонно, понеже $M[i,j]$ не е дефинирано за $j > n$.

```
TEST(M[1 .. n, 1 .. n]: матрица на съседство, i ∈ {1, ..., n})
 1  (* M[k,k] = 0 за 1 ≤ k ≤ n *)
 2  res ← TRUE
 3  for k = 1 to n
 4      if M[i,k] ≠ 0
 5          res ← FALSE
 6  for k = 1 to i − 1
 7      if M[k,i] ≠ 1
 8          res ← FALSE
 9  for k = i + 1 to n
10      if M[k,i] ≠ 1
11          res ← FALSE
12  return res
```

Лема 1: *Алгоритъм* TEST(M, i) *връща* TRUE *тогава и само тогава, когато* M *е матрицата на съседство на граф, в който връх* i *е сръхсифон.*

Доказателство: Приемаме, че твърдението е очевидно вярно. □

Лема 2: *Разглеждаме едно изпълнение на външния* ***while****-цикъл (редове 3–13). Нека в началото на неговата работа е вярно, че* $i = j$, $j \leq n$ *и за всеки* $k \in \{1, \ldots, i-1\}$, *връх* k *не е сръхсифон. Тогава ефектът от работата на вътрешния* ***while****-цикъл (редове 4–5) е следният:*

- *или* $j \leq n$ *и за всеки* $k \in \{1, \ldots, j-1\}$ *е вярно, че връх* k *не е сръхсифон,*
- *или* $j = n + 1$ *и, ако има сръхсифон, той е връх* i.

Доказателство: Следното твърдение е инвариант за вътрешния **while**-цикъл (редове 4–5):

> При всяко достигане на изпълнението на ред 4, за всеки $k \in \{i+1, \ldots, j-1\}$ е вярно, че връх k не е свръхсифон. Освен това, $i \leq j$ и $j \leq n+1$.

База: При първото достигане на ред 4—в рамките на текущото изпълнение на външния цикъл—в сила е $i = j$ и $j \leq n$ е по условие. Тогава множеството $\{i+1, \ldots, j-1\}$ е $\{i+1, \ldots, i-1\}$, което

е празното множество. Ерго, в празния смисъл е вярно, че всеки връх от него не е свръхсифон. Очевидно инвариантът е в сила.

Поддръжка: Да допуснем, че инвариантът е в сила при някое достигане на ред 4, което не е последното. Щом не е последното достигане, изпълнено е $j \leq n$ и $M[i,j] = 0$. Щом $i \leq j$ и $j \leq n$, вярно е, че $[i,j]$ е валиден адрес на клетка от масива.

- Ако $i = j$, след инкрементирането на j на ред 5, множеството $\{i+1,\ldots,j-1\}$ е $\{i+1,\ldots,i\}$, което е празното множество. Ерго, в празния смисъл е вярно, че всеки връх от него не е свръхсифон. Очевидно $i \leq j$ и $j \leq n+1$ също са в сила, така че инвариантът е в сила при следващото достигане на ред 4.

- Ако $i < j$ и $M[i,j] = 0$, връх j със сигурност не е свръхсифон, понеже няма ребро от i към него. От този факт и индуктивното предположение следва, че всеки връх от $\{i+1,\ldots,j\}$ не е свръхсифон. След инкрементирането на j на ред 5, по отношение на новото j, имаме право да твърдим, че всеки връх от $\{i+1,\ldots,j-1\}$ не е свръхсифон. Очевидно $i \leq j$ и $j \leq n+1$ също са в сила, така че инвариантът е в сила при следващото достигане на ред 4.

Терминация: При последното достигане на ред 4 е изпълнено $\neg(j \leq n \wedge M[i,j] = 0)$, което, по закона на De Morgan, е еквивалентно на $j > n \vee M[i,j] \neq 0$. Забележете, че $j > n$ и $M[i,j] \neq 0$ са взаимно несъвместими, така че имаме право да се изразим с изключващо-или: или $j > n$, или $M[i,j] \neq 0$.

- Ако $j > n$, то $j = n+1$, понеже $j \leq n+1$ от инварианта. В този случай от условието на лемата имаме, че нито един връх от $\{1,\ldots,i-1\}$ не е свръхсифон, а от току-що доказания инвариант имаме, че нито един връх от $\{i+1,\ldots,j-1\}$ не е свръхсифон, като последното, предвид $j = n+1$, всъщност е: нито един връх от $\{i+1,\ldots,n\}$ не е свръхсифон. Заключаваме, че ако има свръхсифон, той е връх i.

- Ако $M[i,j] \neq 0$, трябва да е вярно, че $M[i,j] = 1$, понеже матрицата е булева. Но това означава, че връх i не е свръхсифон, щом в ред номер i има единица. Това заключение, заедно с условието, че нито един връх от $\{1,\ldots,i-1\}$ не е свръхсифон, и току-що доказания инвариант, казващ, че нито един връх от $\{i+1,\ldots,j-1\}$ не е свръхсифон, ни дава право да кажем, че нито един връх от $\{i+1,\ldots,j-1\}$ не е свръхсифон.

С което лемата е доказана. □

Теорема: *Алгоритъм* UNIVERSAL SINK($M[1\,..\,n, 1\,..\,n]$) *терминира за всеки вход. Той връща*

- $i \in \{1,\ldots,n\}$*, ако в графа, чиято матрица е* M*, връх* i *е свръхсифон,*
- -1*, ако в графа, чиято матрица е* M*, няма свръхсифон.*

Доказателство: Следното твърдение е инвариант за външния **while**-цикъл (редове 3–13):

> При всяко достигане на изпълнението на ред 3, за всеки $k \in \{1,\ldots,i-1\}$ е вярно, че връх k не е свръхсифон. Освен това, $i = j$ и $j \leq n$.

База: При първото достигане на ред 3, множеството $\{1,\ldots,i-1\}$ е празно, понеже $i = 1$ заради присвояването на ред 2. Поради това, съждението "за всеки $k \in \{1,\ldots,i-1\}$ е вярно, че връх k не е сифон" е вярно в празния смисъл. А $i = j$ и $j \leq n$ заради присвояването на ред 2 ($n \geq 1$, защото поне един връх трябва да има, графи без върхове не разглеждаме).

Поддръжка: Да допуснем, че инвариантът е в сила за някое достигане на ред 3, което не е последното. Щом не е последното, вярно е, че $j \neq n+1$ след изпълнението на вътрешния цикъл; тоест, $j \leq n$ след изпълнението на вътрешния цикъл. От Лема 2 знаем, че след приключването на вътрешния **while**-цикъл,

- или $j \leq n$ и за всеки $k \in \{1, \ldots, j-1\}$ е вярно, че връх k не е свръхсифон,
- или $j = n+1$ и, ако има свръхсифон, той е връх i.

Но при текущите допускания само първото от тези може да се реализира. Щом $j \leq n$ след приключване на вътрешния цикъл, булевото условие на ред 6 е лъжа и изпълнението отива на ред 13, където i става j. Изпълнението отива на ред 3, като инвариантът очевидно остава в сила.

Терминация: Да разгледаме последното достигане на ред 3. Такова трябва да има, защото от Лема 2 знаем, че след приключването на вътрешния цикъл или $j \leq n$, или $j = n+1$; тъй като j се инкрементира поне веднъж при всяко изпълнение на вътрешния цикъл, а има поне едно изпълнение на вътрешния цикъл при всяко изпълнение на външния заради $j \leq n$ (част от инварианта), заключаваме, че има изпълнение на външния цикъл, такова че след изпълнението на вътрешния цикъл имаме $j = n+1$. И така, последно достигане на ред 3 има, сега разглеждаме него и по-точно момента от него след приключването на вътрешния цикъл, в който, както видяхме, $j = n+1$. Съгласно Лема 2, ако има свръхсифон, той може да е само връх i.

Булевото условие на ред 6 е истина и алгоритъмът вика TEST(M, i). Прилагаме Лема 1 и с това приключваме доказателството на теоремата. □

Сложността по време е $O(n)$ по следните причини. Изключвайки работата на помощния алгоритъм TEST, който се вика само един път, всички изпълнения на външния цикъл стават във време $O(n)$ поради строгото нарастване на j при всяка итерация (на външния). От друга страна, TEST работи очевидно във време $O(n)$. Като цяло, сложността е $O(n)$.

Зад. 2 Дадена редица от n реални числа $(x_1, x_2, \ldots, x_n)$, като $x_1 < x_2 < \cdots < x_n$. Предложете колкото е възможно по-ефикасен алгоритъм, който намира множество с минимална мощност от затворени интервали, всеки с дължина единица, които покриват редицата в смисъл, че всяко число от редицата принадлежи на поне един от тези интервали. Анализирайте коректността и сложността по време на Вашия алгоритъм. В тази задача не се иска прецизно доказателство за коректност с инвариант, а само обосновка на оптималността.

Решение: Редицата е сортирана по условие. Следният алчен алгоритъм решава задачата.

```
GREEDY((x_1, ..., x_n), такава че x_1 < ··· < x_n)
 1  S ← ∅
 2  j ← 1
 3  while j ≤ n do
 4      генерирай нов интервал [x_j, x_j + 1]
 5      добави [x_j, x_j + 1] към S
 6      i ← j
 7      while i ≤ n and x_i ≤ x_j + 1 do
 8          i++
 9      j ← i
10  return S
```

Това, че алгоритъмът терминира, че е с линейна сложност по време и че връща множество интервали, което покрива числата на редицата, е очевидно. Остава да покажем, че върнатото множество от интервали е минимално.

Да допуснем, че GREEDY не е оптимален в смисъл, че за поне една редица $(x_1, \ldots, x_n)$, изпълнението на GREEDY$(x_1, \ldots, x_n)$ връща множество от интервали, което не е минимално. Тогава съществува множество S' от единични затворени интервали, което покрива редицата, такова че $|S'| < |S|$.

Разглеждаме произволна редица $(x_1, \ldots, x_n)$, такава че:

- изпълнението на GREEDY$(x_1, \ldots, x_n)$ връща неоптимално множество интервали и
- n е минимално.

Забелязваме, че е невъзможно $S' \subset S$, защото интервалите в S не се пресичат два по два, така че никое число не се покрива от повече от един интервал, и всеки от тях съдържа поне едно число; ерго, ако просто махнем интервал от S, поне едно число ще остане непокрито.

Заключаваме, че S' съдържа поне един интервал, който не се съдържа в S. Да сортираме интервалите в S и интервалите в S' по естествения начин – по началната стойност (или крайната, това дава същата наредба, понеже интервалите са единични). Нека k е минималното число, такова че k-ият интервал в нареденото S е различен от k-ия интервал в нареденото S'. Такова k трябва да има, инак S и S' биха съвпадали. Нека k-ият интервал в S е $[x_i, x_i + 1]$, а k-ият интервал в S' е $[x, x + 1]$. Очевидно $x_i \neq x$, инак тези интервали биха съвпадали.

Ключовото наблюдение е, че $x < x_i$. Причината е следната: алгоритъмът GREEDY е сложил $[x_i, x_i + 1]$ в S поради това, че числото или x_i не се покрива от предния интервал (от S), или $i = 1$. Ерго, ако $x_i < x$, числото x_i би било непокрито от никой интервал от S'; интервалите в S', които са след $[x, x + 1]$ в наредбата, със сигурност не биха покривали x_i, ако $[x, x + 1]$ не покрива x_i; а интервалите в S', които са преди $[x, x + 1]$, съвпадат с интервали от S по конструкция.

И така, $x < x_i$. Да разгледаме

$$S^* = (S' \setminus \{[x, x + 1]\}) \cup \{[x_i, x_i + 1]\}$$

Веднага се вижда, че S^* е допустимо решение, тъй като $[x_i, x_i + 1]$ покрива всички числа, които се покриват от $[x, x + 1]$. Освен това, $|S^*| = |S'|$, така че S^* е оптимално, също като S'.

Да разгледаме числата, които остават непокрити от интервалите до и включително $[x_i, x_i + 1]$ (тези интервали са едни и същи в S и в S^*). Такива числа трябва да има, инак S^* би съвпадало с

S. Но тези оставащи числа представляват екземпляр на задачата, върху който GREEDY не връща оптимално решение, и освен това са по-малко от n на брой. А в нашата конструкция n е минималното число, за което има контрапример с n числа.

Полученото противоречие показва, че алгоритъмът GREEDY винаги намира оптимално решение.

Алгоритъмът очевидно е с линейна сложност по време, понеже разглежда всяко x_i точно един път и извършва само $O(1)$ работа за всяко x_i.

Зад. 3 Направете полиномиална (Karp) редукция от задачата Хамилтонов цикъл към задачата Хамилтонов път. Обосновете редукцията.

Решение: Това е правено на лекции.