Зад. 1 Разгледайте следния алгоритъм.

```
AlgX(A: множество от числа, n = |A|)
 1  Sort(A)    /* резултатът е сортираният масив A[1,...,n] */
 2  c ← 0
 3  for i ← 1 to n − 2
 4      p ← i + 1, q ← n
 5      d ← 0
 6      while p < q do
 7          if A[i] + A[p] + A[q] = 0
 8              d ++
 9              p ++, q −−
10          else if  A[i] + A[p] + A[q] > 0
11              q −−
12          else
13              p ++
14      c ← c + d
15  return c
```

Триада е всяко триелементно подмножество на A, за което сумата от елементите е нула. Примерно, ако входното множество е $\{1, 5, 0, -1, -5, -2, 2, 3\}$, триадите са $\{-1, 0, 1\}$, $\{-2, 0, 2\}$, $\{-5, 0, 5\}$, $\{-5, 2, 3\}$ и $\{-2, -1, 3\}$. Докажете формално и прецизно, че AlgX връща броя на триадите в A.

Решение Ще използваме техниката с инварианта на цикъла. Първо ще споменем три помощни твърдения – те са прекалено очевидни, за да се наричат леми. Нека $A[1, \ldots, n]$ е сортиран и нека $1 \leqslant i < j < k \leqslant n$.

Факт1 Ако $A[i] + A[j] + A[k] \leqslant 0$, то $A[i] + A[j] + A[k'] < 0$ за всяко $k' < k$.

Факт2 Ако $A[i] + A[j] + A[k] \geqslant 0$, то $A[i] + A[j'] + A[k] > 0$ за всяко $j' > j$.

Факт3 Ако $\{A[i], A[j], A[k]\}$ е триада, то $\{A[i], A[j], A[k']\}$ не е триада за всяко $k' < k$ и $\{A[i], A[j'], A[k]\}$ не е триада за всяко $j' > j$.

Факт3 е очевидно следствие на **Факт1** и **Факт2**.

Вътрешният цикъл Първо разглеждаме работата на вътрешния цикъл. По отношение на **едно** изпълнение на външния цикъл разглеждаме **всички** изпълнения на вътрешния цикъл. Ще докажем следното помощно твърдение, което ще наречем **Лема1**: числото d на ред 14 съдържа точно броя на триадите, чиито минимален елемент е $A[i]$. Доказателството на лемата ползва следната инварианта за вътрешния цикъл:

> *при всяко достигане на ред 6,* d *е броят на триадите, в които минималният елемент е* $A[i]$*, средният елемент е от* $A[i + 1, \ldots, p - 1]$*, а максималният елемент е от* $A[q + 1, \ldots, n]$.

Да докажем тази инварианта. **База:** $p = i + 1$ и $q = n$ заради присвояването на ред 4, което означава, че подмасивите $A[i + 1, \ldots, p - 1]$ и $A[q + 1, \ldots, n]$ са съответно $A[i + 1, \ldots, i]$ и $A[n + 1, \ldots, n]$, тоест са празни. Броят на триадите със среден и максимален елемент от празен подмасив е нула. От друга страна, в този момент $d = 0$ заради присвояването на ред 5. Базата е доказана.

Поддръжка: Нека твърдението е вярно за някое достигане на ред 5, което не е последното. Следните две възможности са взаимно изключващи се и са изчерпателни:

- $\{A[i], A[p], A[q]\}$ е триада. Условието на ред 7 е истина и d бива инкрементирано на ред 8. При допускането, че преди това инкрементиране инвариантата е в сила, след това инкрементиране е вярно, че d е броят на триадите, чиито минимален елемент е $A[i]$, чиито среден (по големина) елемент е от $A[i+1, \ldots, p]$ и чиито максимален елемент е от $A[q, \ldots, n]$. Това може да не е абсолютно

очевидно. Обосновката е **Факт3**. Той гарантира, че броят на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p]$ и максимален от $A[q,\ldots,n]$ може да е най-много с единица по голям от броя на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p-1]$ и максимален от $A[q+1,\ldots,n]$.

Така че след инкрементиране на p и декрементиране на q на ред 9 е вярно, че d е броят на триадите с минимален елемент е $A[i]$, в които средният елемент е от $A[i+1,\ldots,p-1]$, а максималният е от $A[q+1,\ldots,n]$. Инвариантата се запазва.

- $\{A[i], A[p], A[q]\}$ не е триада. Следните възможности са взаимно изключващи се и изчерпателни.
  - $A[i] + A[p] + A[q] > 0$. Тогава $A[i] + A[p'] + A[q] > 0$ за $p' \geqslant p$ от **Факт2**. Тогава $A[q]$ не е в триада с $A[i]$ и $A[p']$ за никое $p' \geqslant p$. Тогава броят на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p-1]$ и максимален елемент от $A[q+1,\ldots,n]$ е равен на броя на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p-1]$ и максимален елемент от $A[q,\ldots,n]$.

    От друга страна, условието на ред 10 е истина и изпълнението отива на ред 11, където q се декрементира. Спрямо новото q е вярно, че d е броят на триадите, чиито минимален елемент е $A[i]$, чиито среден елемент е от $A[i+1,\ldots,p-1]$ и чиито максимален елемент е от $A[q+1,\ldots,n]$. Инвариантата се запазва.
  - $A[i]+A[p]+A[q] < 0$. Тогава $A[i]+A[p]+A[q'] < 0$ за $q' \leqslant q$ от **Факт1**. Тогава $A[p]$ не е в триада с $A[i]$ и $A[q]$ за $q' \leqslant q$. Тогава броят на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p-1]$ и максимален елемент от $A[q+1,\ldots,n]$ е равен на броя на триадите с минимален елемент $A[i]$, среден елемент от $A[i+1,\ldots,p]$ и максимален елемент от $A[q+1,\ldots,n]$

    От друга страна, изпълнението отива на ред 13, където p се инкрементира. Спрямо новото p е вярно, че d е броят на триадите, чиито минимален елемент е $A[i]$, чиито среден елемент е от $A[i+1,\ldots,p-1]$ и чиито максимален елемент е от $A[q+1,\ldots,n]$. Инвариантата се запазва.

Терминиране: Изпълнението е на ред 6 и $p \nleqslant q$. Тоест, $p \geqslant q$. Следните две възможности са изчерпателни и взаимно изключващи се. Нека $\hat{p}$ и $\hat{q}$ са съответно стойностите на p и q в началото на последното изпълнение на тялото на вътрешния цикъл.

- При последното изпълнение на тялото на вътрешния цикъл изпълнението е "излязло" през ред 9. Очевидно $\{A[i], A[\hat{p}], A[\hat{q}]\}$ е триада. Съгласно **Факт3**, $A[i]$ и $A[\hat{p}]$ не може да образуват триада с никое $A[q']$ за $q' < \hat{q}$, и освен това, $A[i]$ и $A[\hat{q}]$ не може да образуват триада с никое $A[p']$ за $p' > \hat{p}$. Следователно, независимо дали
  - $\hat{q} - \hat{p} = 1$, при което след изпълнението на ред 9, $A[i+1,\ldots,p-1]$ и $A[q+1,\ldots,n]$ са разбиване на $A[i+1,\ldots,n]$,
  - или $\hat{q} - \hat{p} = 2$, при което след изпълнението на ред 9, $A[i+1,\ldots,p-1]$ и $A[q+1,\ldots,n]$ не са разбиване на $A[i+1,\ldots,n]$, защото $A[p] = A[q]$ остава извън тях,

  d съдържа броя на триадите с минимален елемент $A[i]$. Във втората възможност, $A[p] = A[q]$ не може да е в триада с минимален елемент $A[i]$ и трети елемент нито от $A[i+1,\ldots,p-1]$, нито от $A[q+1,\ldots,n]$.
- При последното изпълнение на тялото на вътрешния цикъл изпълнението е "излязло" през ред 11 или ред 13. Вярно е, че $\hat{p} = \hat{q} - 1$, а след изпълнението на ред 11 или ред 13 е вярно, че $p = q$. Тогава $A[i+1,\ldots,p-1]$ и $A[q+1,\ldots,n]$ не са разбиване на $A[i+1,\ldots,n]$, защото $A[p] = A[q]$ остава непокрит.

  Съгласно **Факт2**, ако излизането е станало през ред 13, или **Факт1**, ако излизането е станало през ред 11, елементът $A[p] = A[q]$ и в двата случая не може да участва в триада с минимален елемент $A[i]$, която не е преброена в d.

С това доказателството на **Лема1** приключва.

Външният цикъл Инвариантa за външния цикъл е: при всяко достигане на ред 3, променливата c съдържа броя на триадите, чийто минимален елемент се намира в $\mathsf{A}[1, \ldots, \mathsf{i}-1]$. Базата е $\mathsf{i} = 1$ при първото достигане. Тогава, от една страна броят на тези триади е нула, понеже $\mathsf{A}[1, \ldots, \mathsf{i}-1]$ е празен, а от друга страна c съдържа нула заради ред 2. Поддръжката се доказва лесно: ако твърдението е вярно за някое достигане, което не е последното, то съгласно **Лема1** присвояването на ред 14 добавя към c точно броя на триадите с минимален елемент $\mathsf{A}[\mathsf{i}]$, така че след инкрементирането на i на ред 3, инвариантата е в сила буквално. Да разгледаме последното достигане на ред 3. Тогава $\mathsf{i} = \mathsf{n}-1$ и получаваме, че c съдържа броя на триадите, чийто минимален елемент се намира в $\mathsf{A}[1, \ldots, \mathsf{n}-2]$. Но това са всички триади. Тогава на ред 15 алгоритъмът връща броя на всички триади.

Зад. 2 Подредете по асимптотично нарастване следните шест функции:

$$f_1(n) = n^n \qquad f_2(n) = (n+1)^n \qquad f_3(n) = e^n \cdot n!,$$
$$f_4(n) = \sum_{k=1}^{n} k! \qquad f_5(n) = \left(\sum_{k=0}^{n} \binom{n}{k}\right)^4 \qquad f_6(n) = \binom{4n}{2n}$$

Решение Ще докажем, че $f_5(n) \prec f_6(n)$. От Дискретни структури знаем, че $\sum_{k=0}^{n} \binom{n}{k} = 2^n$, откъдето заключаваме, че $f_5(n) = (2^n)^4 = 16^n$. От лекции знаем, че

$$\binom{n}{\frac{n}{2}} \asymp \frac{1}{\sqrt{n}} 2^n$$

Тогава

$$f_6(n) = \binom{4n}{2n} \asymp \frac{1}{\sqrt{4n}} 2^{4n} \asymp \frac{1}{\sqrt{n}} 16^n$$

Оттук следва, че

$$f_6(n) \prec f_5(n)$$

Ще сравним f_5 с f_4. Тривиално се показва, че $f_4(n) \asymp n!$; тоест, най-голямото събираемо, което е $n!$, дава асимптотиката на сумата. Това може да се направи по много начини. Може по индукция по n да се докаже, че първото събираемо е по-голямо от сумата на останалите, за всички достатъчно големи n. Може да се извади $n!$ пред скоби, като дробите вътре, може би с изключение на константен брой от тях, имат сума, ограничена от константа, понеже съответният ред е сходим.

Но $a^n \prec n!$ за всяка $a > 1$, което е напълно очевидно. Тогава $16^n \prec n!$. Тогава $f_5(n) \prec f_4(n)$. Засега имаме

$$f_6(n) \prec f_5(n) \prec f_4(n)$$

Да сравним f_4 и f_1. Но напълно очевидно е, че $n! \prec n^n$, откъдето $f_4(n) \prec f_1(n)$. Засега имаме

$$f_6(n) \prec f_5(n) \prec f_4(n) \prec f_1(n)$$

Ще докажем, че $f_2(n) \asymp f_1(n)$. Наистина,

$$\lim_{n\to\infty} \frac{(n+1)^n}{n^n} = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = e$$

Засега имаме

$$f_6(n) \prec f_5(n) \prec f_4(n) \prec f_1(n) \asymp f_2(n)$$

Да сравним f_1 с f_3. Прилагаме апроксимацията на Stirling върху $f_3(n)$ и получаваме

$$f_3(n) \asymp e^n \cdot \sqrt{n} \frac{n^n}{e^n} = \sqrt{n} \cdot n^n$$

Очевидно $f_1(n) \prec f_3(n)$. Окончателната наредба е

$$f_6(n) \prec f_5(n) \prec f_4(n) \prec f_1(n) \asymp f_2(n) \prec f_3(n)$$

Зад. 3 Дадено е множество от наредени двойки $X = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, такива че $a_i < b_i$ за $1 \leqslant i \leqslant n$. Известно е, че числата a_1, $\ldots$, a_n, b_1, $\ldots$, b_n са две по две различни. За всеки две наредени двойки (a_i, b_i) и (a_j, b_j), където $i \neq j$, казваме, че са *независими*, ако е изпълнено $a_i < b_i < a_j < b_j$ или $a_j < b_j < a_i < b_i$. Ако е известно, че наредените двойки (a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2}), $\ldots$, (a_{i_k}, b_{i_k}) са две по две независими, казваме, че масивът от тях

$$[(a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2}), \ldots, (a_{i_k}, b_{i_k})]$$

е *растящ*, ако

$$a_{i_1} < b_{i_1} < a_{i_2} < b_{i_2} < \cdots < a_{i_k} < b_{i_k}$$

Професор Интервалски твърди, че разполага с алгоритъм, който във време $O(n \lg \lg n)$ намира множество $Y \subseteq X$, такова че наредените двойки в Y са две по две независими и $|Y| \geqslant |Z|$ за всяко $Z \subseteq X$, такова че наредените двойки в Z са независими. Нещо повече, професорът твърди, че неговият алгоритъм връща Y в растящ масив. Опровергайте го: докажете, че такъв алгоритъм няма.

Решение Това е правено на лекции. Задачата е Interval Scheduling – просто тук интервалите са наречени "наредени двойки". Ако за тази задача има алгоритъм със сложност $O(n \lg \lg n)$, той може да се ползва като процедура за алгоритъм за сортиране със сложност $O(n \lg \lg n)$. Ето как: при даден вход-редица от цели числа $(a_1, \ldots, a_n)$, две по две различни, генерираме съответна редица от наредени двойки $((a_1, a_1 + \epsilon), \ldots, (a_n, a_n + \epsilon))$, където ϵ е положително число, по-малко от единица; да кажем $\epsilon = 0.1$. Пускаме хипотетичния алгоритъм на професора върху редицата от наредени двойки. Тъй като нито две от тях не се настъпват, иначе казано, всеки две от тях са независими, максималното подмножество от две по две независими двойки е самото множество от всички тях. Нещо повече – алгоритъмът на професора би върнал редица от тях в нарастваща големина на първите елементи, които са оригиналните числа. От тази редица е тривиално в линейно време да се генерира сортираната редица, съответстваща на входната $(a_1, \ldots, a_n)$.

И така, ако има такъв алгоритъм, има сортиране във време $O(n \lg \lg n)$. Това директно противоречи на долната граница $\Omega(n \lg n)$ за сортиране на произволни числа.

Зад. 4 Нека $G = (V, E)$ е неориентиран обикновен свързан тегловен граф, като $w : E \to \mathbb{R}^+$ е тегловната функция. Нека w е инекция. Докажете, че за всеки два върха $u, v \in V$, Prim(G, w, u) и Prim(G, w, v) намират дърветата, които са изоморфни. "Prim"означава алгоритъма на Prim за намиране на МПД.

Решение Тегловната функция да е инекция е същото като теглата на ребрата да са уникални. Ще докажем, че ако теглата са уникални, то минималното покриващо дърво е само едно. Да го наречем T. Това решава задачата: както и да избираме началния връх за Prim, както и да са списъците на съседство, алгоритъмът ще намери именно T. Релацията на изоморфизъм е очевидно рефлексивна и T е изоморфно на себе си.

И така, задачата се свежда до това да се докаже, че МПД е едно единствено. Това, че съществува МПД, тривиално следва от факта, че G е свързан. Да допуснем, че има поне две различни МПД-та T_1 и T_2, въпреки че w е инекция. Тогава в поне едното от тях има ребро, което не се съдържа в другото (дори по-силно твърдение е вярно: тъй като броят на ребрата им е еднакъв, всяко от тях има ребро, което не се съдържа в другото). Нека e' е реброто с **минимално** тегло, което се съдържа в точно едно от T_1 и T_2. БОО, нека e' е ребро от T_1. Щом w е инекция, e' е добре дефинирано.

Да разгледаме графа $U = (V, E(T_2) \cup \{e'\})$ с тегло $w(U) = w(T_2) + w(e')$. Както знаем от лекции, той е уницикличен граф. Да кажем, че цикълът е c. Очевидно c съдържа ребро $e'' \in E(T_2)$, такова че $e'' \neq e'$ и $e'' \notin T_1$. Но тогава е вярно, че $w(e') < w(e'')$; да си припомним как дефинирахме e'.

Както знаем от Дискретни структрури, $U - e''$ е дърво, и то покриващо дърво. Неговото тегло е $w(T_2) + w(e') - w(e'')$. Но $w(e') - w(e'') < 0$. Тогава това е покриващо дърво с тегло, по-малко от $w(T_2)$, в противоречие с прежде направеното допускане, че T_2 е МПД.

Зад. 5 Дадена е група от n човека. Всеки от тях има или няма някакви жетони. Всички жетони са еднакви. Нещо повече: известно е, че за някакво k, всеки човек има между 0 и k жетона включително.

Предложете колкото е възможно по-ефикасен алгоритъм, който разбива групата на две подгрупи (подмножества), такива че сумите от жетоните на хората в едната и в другата група са максимално близки. Иначе казано, ако в едната група имат сумарно s_1 жетона, а в другата имат сумарно s_2 жетона, разбиването да е такова, че $|s_1 - s_2|$ да е минимална. Съвсем накратко анализирайте коректността и сложността на алгоритъма си.

Достатъчно е Вашият алгоритъм да намира минималната разлика като число. Не е необходимо да се генерира самото разбиване.

Класифицирайте алгоритъма си като сложност по време. Има се предвид следното: на лекции сме разглеждали полиномиални алгоритми, споменавали сме експоненциални алгоритми, говорили сме и за други видове алгоритми в този смисъл. Каква бихте нарекли сложността по време на Вашия алгоритъм от тази задача?

Решение Да кажем, че всички жетони са S на брой. Очевидно $S \leqslant nk$. За да са реши задачата е достатъчно да се намери максималното число X, такова че $X \leqslant \lfloor S/2 \rfloor$ и съществува подмножество от хора, които сумарно имат X жетона. Тогава останалите хора имат $S - X$ жетона и цената на решението става $S - 2X$. По-добро решение не може да има, защото съществуването на такова влече съществуване на $X' > X$, такова че $X' \leqslant \lfloor S/2 \rfloor$ и съществува подмножество от хора, които сумарно имат X' жетона.

Задачата прилича много на разглежданата на лекции задача Set Partition – хората са елементите, а броят на жетоните в даден човек е размерът на съответния елемент. Има и разлика обаче. Set Partition е задача за разпознаване, докато тази задача е оптимизационна. Тази задача е и по-обща, понеже алгоритъм за нея тривиално може да се ползва и за Set Partition: ако намерената цена е нула, то примерът е ДА-пример за Set Partition, в противен случай е НЕ-пример за Set Partition.

От друга страна, тази задача е частен случай на задачата за раницата в 0-1 варианта. Хората от тази задача отговарят на предметите от задачата за раницата. Хората тук са уникални, точно както 0-1 варианта на задачата за раницата предметите не се повтарят. Цената на предмет, както и размерът на предмет, е броят на жетоните в съответния човек. Капацитетът на раницата е $C = \lfloor S/2 \rfloor$. Нашият алгоритъм пуска алгоритъма за раницата върху така построения пример. Последният връща някакво число $W \leqslant C$, при което нашият алгоритъм за Задача 5 връща $S - 2W$.

Алгоритъм със сложност по време за задачата за раницата $O(nC)$ е разгледан на лекции. В термините на тази задача, сложността е $O(n^2k)$, понеже $S = O(nk)$, а $C = \lfloor S/2 \rfloor$.

Алгоритъмът е с псевдополиномиална сложност – понятие, въведено на лекции. $O(n^2k)$ не е полиномиална (в асимптотичния смисъл) горна граница заради k. k по определение е някакво число, което е част от примера. В нашия изчислителен модел, всички числа имат размер единица независимо от големината си. За да бъде $O(n^2k)$ полиномиална горна граница, трябва големината на k да е най-много полиномиална в размера на примера, който на свой ред е $\Theta(n)$; тоест, трябва да има константа d, такава че $k = O(n^d)$. Такова нещо не се казва в условието и нямаме право да го допускаме. Псевдополиномиална сложност в този случай означава, че ако k е записано в унарна бройна система, то сложността е полиномиална. За израза "n^2k" това е вярно. В по-реалистичен изчислителен модел, в който се отчита истински размера на k и k е представено в бинарна система, алгоритъмът има експоненциална сложност по време в най-лошия случай ($k = \Omega(2^n)$). Формално говорейки, не е грешка да се каже, че сложността е недефинирана в нашия изчислителен модел, или че е експоненциална в по-реалстичния модел, отчитащ размера на k, но най-смислено и информативно е "сложността по време е псевдополиномиална".

Зад. 6 Дадено е множество от n на брой d-мерни кутии, всяка от които е d-мерен правоъгълен паралелепипед. За целите на тази задача може да смятате, че всяка кутия е наредената d-орка и че $d \geqslant 2$.

Казваме, че кутия $(x_1, x_2, \ldots, x_d)$ *влиза* в кутия $(y_1, y_2, \ldots, y_d)$, ако има пермутация

$$\pi : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\},$$

такава че $x_1 < y_{\pi(1)}$, $x_2 < y_{\pi(2)}$, $\ldots$, $x_d < y_{\pi(d)}$.

- Докажете, че влизането на кутия в кутия е транзитивна релация.
- Покажете как във време $O(d\sqrt{d})$ може да проверим дали една кутия влиза в друга кутия.
- Предложете ефикасен алгоритъм, който намира максимално дълга редица $\langle b_{i_1}, b_{i_2}, \ldots, b_{i_k} \rangle$, такава че b_{i_j} влиза в $b_{i_{j+1}}$, за $1 \leqslant j < k$. Съвсем накратко обосновете коректността на Вашия алгоритъм и анализирайте сложността му по време.

Забележете, че условието за влизане е по-сложно от $x_1 < y_1$, $x_2 < y_2$, $\ldots$, $x_d < y_d$. Да разгледаме пример за това при $d = 3$. Нека първата кутия е $(2, 3, 2)$, а втората е $(2.1, 2.1, 4)$. Втората дименсия на първата кутия, тоест 3, е по-голяма от втората дименсия на втората кутия, тоест 2.1, поради което не е вярно, че $2 < 2.1$ и $3 < 2.1$ и $2 < 4$. Но ако това са физически кутии, можем да сложим първата кутия във втората, като първо я завъртим и тя стане $(2, 2, 3)$. Сега вече е ясно, че тя може да влезе в кутия $(2.1, 2.1, 4)$, тъй като $2 < 2.1$ и $2 < 2.1$ и $3 < 4$.

Решение Ще докажем транзитивността на влизането на една кутия в друга кутия. Да допуснем, че кутия $(x_1, x_2, \ldots, x_d)$ влиза в кутия $(y_1, y_2, \ldots, y_d)$, а кутия $(y_1, y_2, \ldots, y_d)$ влиза в кутия $(z_1, z_2, \ldots, z_d)$. По определение това означава, че има пермутации

$$\pi : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\}$$
$$\psi : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\}$$

такива че

$$x_1 < y_{\pi(1)}, x_2 < y_{\pi(2)}, \ldots, x_d < y_{\pi(d)}$$
$$y_1 < z_{\psi(1)}, y_2 < z_{\psi(2)}, \ldots, y_d < z_{\psi(d)}$$

Нека μ е композицията $\psi \circ \pi$. Тогава от транзитивността на релацията "$<$" следва, че

$$x_1 < z_{\mu(1)}, x_2 < z_{\mu(2)}, \ldots, x_d < z_{\mu(d)}$$

Тогава кутия $(x_1, x_2, \ldots, x_d)$ влиза в кутия $(z_1, z_2, \ldots, z_d)$. Ерго, релацията на влизане на кутия в кутия е транзитивна.

Проверката за това, дали кутия b_i влиза в кутия b_j е елементарна. Сортират се наредените d-орки на двете кутии във време $O(d \lg d)$ и после във време $O(d)$ се проверява покомпонентно дали компонентите на първата d-орка са по-малки от тази на втората. b_i влиза в кутия b_j тстк отговорът е ДА. Коректността на това следва веднага от определението на задачата Сортиране.

Да разгледаме задачата за намиране на максимална редица от вложени кутии. Едно възможно решение е следното. Първо във време $O(nd \lg d)$ се сортират d-орките на всички кутии. После се сравняват всяка кутия с всяка друга и се отбелязва дали едната от тях влиза в другата. Невъзможно е една кутия да влиза в друга и обратно (релацията на влизане е антисиметрична), така че изходът от всяко сравнение е точно едно от трите:

- първата влиза във втората,
- втората влиза в първата,
- нито една не влиза в другата.

От тези сравнения построяваме ориентиран граф на влизането, чиито върхове са кутиите, а ребро от връх u към връх v има тстк кутията, съответна на u, влиза в кутията, съответна на v. Построяването на графа става във време $O(n^2 d)$.

Очевидно графът е даг – релацията на влизане е и антирефлексивна. Това, което търсим, е най-дълъг път в този даг. Обосновката на това е, че всеки път в този даг съответства на редица от вложени кутии, така че най-дълъг път ни дава максимална редица от вложени кутии, каквато се търси в задачата.

От лекции знаем как се намира най-дълъг път в даг във време, линейно в размера на дага. Размерът на този даг е $O(n^2)$, така че във време $O(n^2)$ намираме желания най-дълъг път и връщаме съответната редица от влагащи се една в друга кутии.

Общата сложност е $O(n^2 d + d \lg d)$. Забележете, че никъде не е казано, че d е константа или дори че $d \leqslant n$.