Примерни решения на домашно № 2

Задача 1: Нека A е двумерен масив $n \times n$, чиито елементи са от $\mathbb{Z} \cup \{\infty\}$. Казваме, че A е *квазисортиран*, ако във всеки ред елементите са сортирани отляво надясно и във всяка колона елементите са сортирани отгоре надолу. Елемент ∞ се интерпретира като празно място – дупка, в която няма число. За целите на домашното, релациите $<$ и $\leqslant$ са дефинирани над $\mathbb{Z} \cup \infty$: вярно е, че

- $\forall k \in \mathbb{Z} : k < \infty$ и $\forall k \in \mathbb{Z} : k \leqslant \infty$,
- $\infty \nless \infty$ и $\infty \leqslant \infty$.

Заради квазисортировката, ако в някой ред има елементи ∞, то те са в края, и ако в някоя колона има елементи ∞, то те са най-долу. Ето пример за квазисортиран масив 5×5:

1	20	25	25	26
1	22	25	30	40
15	30	40	45	∞
16	40	50	150	∞
20	200	2000	∞	∞

Мултиножеството от числата на квазисортиран масив е мултимножеството от тези негови елементи, които не са ∞; елементът ∞ не е число. В показания пример, мултимножеството от числата е

$$\{1, 1, 15, 16, 20, 20, 22, 25, 25, 25, 26, 30, 30, 40, 40, 40, 45, 50, 150, 200, 2000\}_M$$

Квазисортиран масив е *непълен*, ако съдържа поне една стойност ∞, а в противен случай е *пълен*. Масивът от примера е непълен.

1 т. 1. Предложете алгоритъм с колкото е възможно по-добра асимптотична сложност по време, който връща минимума на квазисортиран масив. Каква е сложността по време на Вашия алгоритъм? Обосновете я.

9 т. 2. Предложете алгоритъм с колкото е възможно по-добра асимптотична сложност по време, който връща минимума на квазисортиран масив и го премахва от масива. Това ще рече, че минимумът се премахва от мултимножеството на числата на масива, но масивът остава квазисортиран.

Обосновете съвсем накратко коректността на Вашия алгоритъм. Не е необходимо подробно доказателство, а само формулиране на коректно твърдение.

Каква е сложността по време на Вашия алгоритъм? Обосновете я.

10 т. 3. Предложете алгоритъм с колкото е възможно по-добра асимптотична сложност по време, който добавя число към непълен квазисортиран масив. Да са добави ново число ℓ към непълен квазисортиран масив $n \times n$, чието мултимножество от числа е $\{s_1, \ldots, s_m\}_M$,

означава да се преобразува в квазисортиран масив, чието мултимножество от числа е $\{s_1, \ldots, s_m\}_M \cup \{\ell\}_M$.

Обосновете съвсем накратко коректността на Вашия алгоритъм. Не е необходимо подробно доказателство, а само формулиране на коректно твърдение.

Каква е сложността по време на Вашия алгоритъм? Обосновете я.

10 т. 4. Без да използвате каквито и да е сортиращи алгоритми като процедури, покажете как да генерираме сортираната редица от елементите на даден $n \times n$ квазисортиран масив във време $O(n^3)$, използвайки само алгоритми и резултати от горните подусловия на тази задача.

Обосновете съвсем накратко коректността на Вашия алгоритъм. Не е необходимо подробно доказателство, а само формулиране на коректно твърдение.

Каква е сложността по време на Вашия алгоритъм? Обосновете я.

Решение: Такъв двумерен масив е известен като *Young tableau*, в множествено число *Young tableaux*. Често срещаната дефиниция на "Young tableau" е привидно различна: според нея, Young tableau се състои от редове от клетки, подравнени вляво, като дължините на редовете са в ненарастващ ред отгоре надолу, а числата в клетките са сортирани по редове отляво надясно и по колони отгоре надолу. Двете дефиниции са практически еквивалентни. Ако в стандартната дефиниция допълним редовете с ∞, така че да станат с еднакви дължини, ще получим правоъгълна $m \times n$ таблица, която на свой ред може да допълним с ∞ до квадратна таблица, сортирана хоризонтално и вертикално.

Има известна прилика между квазисортиран $n \times n$ масив и двоична пирамида тип min с размер n^2. Ако си представим квазисортирания масив като едномерен масив $B[1, \ldots, n^2]$ (в компютърната памет той би бил реализиран на ниско ниво като едномерен, а не двумерен масив, защото, хардуерно погледнато, паметта на компютъра е едномерна), иска се подмасивите $B[1, \ldots, n]$, $B[n+1, \ldots, 2n]$, $\ldots$, $B[n^2-n+1, \ldots, n^2]$ да са сортирани, а също така и подмасивите $B[1, n+1, \ldots, n^2-n+1]$, $B[2, n+2, \ldots, n^2-n+2]$, $\ldots$, $B[n, 2n, \ldots, n^2]$ да са сортирани. При min пирамидата се иска сортиране в ненамаляващ ред на поредиците от елементи, отговарящи на пътищата от корена (елемент 1) до листата.

1. Минималният елемент на квазисортиран масив $A[1, \ldots, n][1, \ldots, n]$ е $A[1][1]$. Следният алгоритъм връща минимума на квазисортиран масив и има оптимална сложност по време $\Theta(1)$:

```
QUASI-SORTED-MIN(A[1,...,n][1,...,n]: квазисортиран масив)
  1  return A[1][1]
```

2. Съхраняваме $A[1][1]$ в някаква променлива, записваме ∞ в клетка $[1][1]$ и възстановяме квазисортировката по начин, подобен на MIN-HEAPIFY при двоичните пирамиди: стартирайки от елемент $[1][1]$, докато елементът долу е по-малък или елементът вдясно е по-малък, разменяме текущия елемент с по-малкия от тези двата. Кодът, реализиращ тази идея, е по-къс, ако има sentinel values ∞, разположени по такъв начин, че винаги да има елемент долу и елемент вдясно. С други думи, масивът е $(n+1) \times (n+1)$, като елементите на най-долния ред и в най-дясната колона са само ∞.

QUASI-SORTED-EXTRACT-MIN($A[1,\ldots,n+1][1,\ldots,n+1]$: квазисортиран масив, последният ред и най-дясната колона са ∞)

```
 1  tmp ← A[1][1]
 2  A[1][1] ← ∞
 3  ⟨j, k⟩ ← ⟨1, 1⟩
 4  while A[j][k] > A[j][k + 1]  or A[j][k] > A[j + 1][k] do
 5      if A[j][k] > A[j][k + 1]  and A[j][k] > A[j + 1][k]
 6          if A[j][k + 1] < A[j + 1][k]
 7              swap(A[j][k], A[j][k + 1])
 8              k++
 9          else
10              swap(A[j][k], A[j + 1][k])
11              j++
12      else
13          if A[j][k] > A[j + 1][k]
14              swap(A[j][k], A[j + 1][k])
15              j++
16          else
17              swap(A[j][k], A[j][k + 1])
18              k++
19  return tmp
```

Удобно е да дефинираме понятие, аналогично на "min пирамидална инверсия". "Min пирамидална инверсия" е наредена двойка $\langle i, j\rangle$ в масив $A[1,\ldots,n]$, такава че $1 \leqslant i < j \leqslant n$, $A[i] > A[j]$ и $A[i]$ е предшественик на $A[j]$. В двумерен масив $A[1,\ldots,n][1,\ldots,n]$ можем да дефинираме *Young инверсия* така: това е наредена двойка от наредени двойки $\langle\langle j, k\rangle, \langle j', k'\rangle\rangle$, такава че $1 \leqslant j \leqslant j' \leqslant n$ и $1 \leqslant k \leqslant k' \leqslant n$ и $\langle j, k\rangle \neq \langle j', k'\rangle$ и $A[j][k] > A[j'][k']$. Също както едномерният масив е min пирамида тстк няма min пирамидални инверсии, двумерният масив е Young tableau тстк няма Young инверсии.

Инвариант за **while** цикъла на ред 4 е: при всяко достигане на ред 4, всички Young инверсии в $A[1,\ldots,n][1,\ldots n]$ се намират в подмасива $A[j,\ldots,n][k,\ldots,n]$.

Сложността на алгоритъма е $\Theta(n)$ в най-лошия случай, понеже **while**-цикълът се изпълнява най-много $\Theta(n)$ пъти, понеже индексите j и k само могат да нарастват и наредената двойка $\langle j, k\rangle$ може да приема по-малко от $2n$ различни стойности.

Алгоритъмът може да се реализира и рекурсивно, при което сложността по време се описва от рекурентното уравнение $T(n) = T(n-1) + 1$, чието решение е $T(n) = \Theta(n)$.

3. И тук решението е алгоритъм, доста подобен на аналогичния алгоритъм за добавяне на елемент в двоична пирамида. Намираме първия ред, в който има елемент ∞ (щом масивът е непълен, такъв ред има), слагаме новия елемент k на мястото най-левия ∞ в този ред и разменяме k с този елемент измежду елементите отгоре и вляво, който е не по-малък от другия, докато е необходимо. Все пак има съществена разлика с пирамидите – там има само един елемент, с който да разменяме новия и това е родителят, докато при квазисортираните масиви в общия случай има два предшественика – този вляво и този отгоре. Тук няма нужда от сентинели.

QUASI-SORTED-INSERT($A[1,\ldots,n][1,\ldots,n]$: непълен квазисортиран масив, число ℓ)

```
1  j ← минималният номер на ред, в който има ∞
2  k ← минималният номер на колона, такава че A[j][k] = ∞
3  A[j][k] ← ℓ
4  while (j ⩾ 1 and A[j − 1][k] > A[j][k]) or (k ⩾ 1 and A[j][k − 1] > A[j][k]) do
5      if (j ⩾ 1 and A[j − 1][k] > A[j][k]) and (k ⩾ 1 and A[j][k − 1] > A[j][k])
6          if A[j − 1][k] > A[j][k − 1]
7              swap(A[j][k], A[j − 1][k])
8              j--
9          else
10             swap(A[j][k], A[j][k − 1])
11             k--
12     else
13         if j ⩾ 1 and A[j − 1][k] > A[j][k]
14             swap(A[j][k], A[j − 1][k])
15             j--
16         else
17             swap(A[j][k], A[j][k − 1])
18             k--
```

Инвариант за **while** цикъла на ред 4 е: при всяко достигане на ред 4, всички Young инверсии в $A[1,\ldots,n][1,\ldots n]$ се намират в подмасива $A[1,\ldots,j][1,\ldots,k]$.

Сложността по време в най-лошия случай е $\Theta(n)$, което показваме по начин, напълно аналогичен на предното доказателство за сложност по време.

4. Алгоритъмът е нещо като HEAPSORT, само че не върху двоична пирамида, а върху квазисортиран масив. Няма изискване за константна сложност по памет, така че може да си позволим да ползваме допълнителен масив $D[1,\ldots,n^2]$, в който да разположим сортираните елементи-числа на входния $A[1,\ldots,n][1,\ldots,n]$.

QUASI-SORTED-SORT($A[1,\ldots,n][1,\ldots,n]$: квазисортиран масив)

```
1  създай масив D[1, . . . , n^2], инициализирайки го с ∞
2  for i ← 1 to n^2
3      D[i] ← QUASI-SORTED-EXTRACT-MIN(A)
4  return D
```

Инвариант за **for** цикъла е: при всяко достигане на ред 3, подмасивът $D[1,\ldots,i-1]$ се състои от $i-1$ на брой най-малки елементи A в сортиран вид. Освен това, A е квазисортиран.

Сложността по време е $\Theta(n^3)$. Цикълът се изпълнява $\Theta(n^2)$ пъти, а всяко викане на QUASI-SORTED-EXTRACT-MIN има сложност по време $O(n)$, откъдето имаме асимптотична горна граница $O(n^3)$ за целия алгоритъм. За да получим и асимптотична долна граница $\Omega(n^3)$ за целия алгоритъм, достатъчно е да съобразим, че е възможно да има елементи във входния A, чийто брой е $\Theta(n^2)$, всеки от които трябва "пропътува" $\Theta(n)$ размени в QUASI-SORTED-EXTRACT-MIN.

Задача 2: Решете следните шест рекурентни уравнения.

$$A(n) = 7A\left(\frac{n}{\sqrt{3}}\right) + n\lg n \qquad B(n) = B\left(\frac{2n}{9}\right) + B\left(\frac{n}{7}\right) + B\left(\frac{n}{2}\right) + n$$

$$C(n) = 3C(n-1) + n2^n \qquad R(n) = 6R(n-1) + 11R(n-2) + n^{\sqrt{n}}$$

$$S(n) = 4S\left(\frac{n}{2}\right) + n^2(\lg n)^3 \qquad T(n) = 4T\left(\frac{n}{2}\right) + n^{\lg n}$$

Решение: Решаваме $A(n) = 7A\left(\frac{n}{\sqrt{3}}\right) + n\lg n$. Пригаме МТ: $a = 7$, $b = \sqrt{3}$, оттук $\log_b a = \log_{\sqrt{3}} 7 \approx 3.542487498$. Точната стойност няма значение. Важното е, че е по-голяма от 1. Тогава $n\lg n = O(n^{\log_b a}/n^\epsilon)$, за някое $\epsilon > 0$. Тогава първият случай на МТ е приложим и съгласно него, $A(n) \asymp n^{log_{\sqrt{3}}7}$.

Решаваме $B(n) = B\left(\frac{2n}{9}\right) + B\left(\frac{n}{7}\right) + B\left(\frac{n}{2}\right) + n$. МТ не е приложима. Тъй като $\frac{2}{9} + \frac{1}{7} + \frac{1}{2} = \frac{109}{126}$, което е по-малко от 1, интуитивно е ясно, че $B(n)$ е асимптотично еквивалентно на нехомогенната част n^1. Ще го докажем по индукция. Първо ще докажем, че $B(n) = O(n)$. По определение, това е същото като да докажем, че има положителна константа c, такава че $B(n) \leqslant cn$ за всички достатъчно големи стойности на n. Използваме силна индукция. Допускаме, че

$$B\left(\frac{2n}{9}\right) \leqslant c\left(\frac{2n}{9}\right)$$
$$B\left(\frac{n}{7}\right) \leqslant c\left(\frac{n}{7}\right)$$
$$B\left(\frac{n}{2}\right) \leqslant c\left(\frac{n}{2}\right)$$

Тогава

$$B(n) \leqslant c\left(\frac{2n}{9}\right) + c\left(\frac{n}{7}\right) + c\left(\frac{n}{2}\right) + n$$

което е същото като

$$B(n) \leqslant \frac{109cn}{126} + n$$

Това извеждаме от индукционните предположения и дефиницията на $B(n)$. Дали има константа $c > 0$, такава че

$$\frac{109cn}{126} + n \leqslant cn \;\leftrightarrow\; \frac{109c}{126} + 1 \leqslant c \;\leftrightarrow\; 109c + 126 \leqslant 126c \;\leftrightarrow\; 126 \leqslant 17c$$

Да, всяка положителна константа c, по-голяма от $\frac{126}{17}$, върши работа за доказателството. Доказахме, че $B(n) = O(n)$. Ще докажем, че $B(n) = \Omega(n)$. Това е доста очевидно предвид факта, че $B(n)$ се дефинира чрез израз, съдържащ $+n$ и още едно положително събираемо, но да го докажем строго формално по индукция. По определение, $B(n) = \Omega(n)$ е същото като да има положителна константа d, такава че $B(n) \geqslant dn$ за всички достатъчно големи n. Допускаме, че

$$B\left(\frac{2n}{9}\right) \geqslant d\left(\frac{2n}{9}\right)$$
$$B\left(\frac{n}{7}\right) \geqslant d\left(\frac{n}{7}\right)$$
$$B\left(\frac{n}{2}\right) \geqslant d\left(\frac{n}{2}\right)$$

Тогава

$$B(n) \geqslant d\left(\frac{2n}{9}\right) + d\left(\frac{n}{7}\right) + d\left(\frac{n}{2}\right) + n$$

което е същото като

$$B(n) \geqslant \frac{109dn}{126} + n$$

Това извеждаме от индукционните предположения и дефиницията на $B(n)$. Дали има константа $d > 0$, такава че

$$\frac{109dn}{126} + n \geqslant dn \;\leftrightarrow\; \frac{109d}{126} + 1 \geqslant d \;\leftrightarrow\; 109d + 126 \geqslant 126d \;\leftrightarrow\; 126 \geqslant 17d$$

Всяка положителна константа d, по-малка от $\frac{126}{17}$, върши работа за доказателството.

Решаваме $C(n) = 3C(n-1) + n2^n$. Използваме метода с характеристичното уравнение. Характеристичното уравнение е $x - 3 = 0$ с мултимножество от корените $\{3\}_M$. От нехомогенната част имаме мултимножество $\{2, 2\}_M$, тъй като полиномът е от първа степен. Обединяваме двете мултимножества и получаваме $\{2, 2, 3\}$. Съгласно изучаваното на лекции, $C(n) \asymp 3^n$.

Решаваме $R(n) = 6R(n-1) + 11R(n-2) + n^{\sqrt{n}}$. Това уравнение не може да се реши с метода с характеристичното уравнение, понеже нехомогенната част не е от желания вид: $n^{\sqrt{n}}$ не е произведение от полином на n и експоненциална функция на n. Ще решим уравнението с индукция, но преди това трябва да сме наясно какво ще доказваме по индукция. Хомогенното уравнение е $R(n) = 6R(n-1) + 11R(n-2)$ с решение $R(n) \asymp (3 + 2\sqrt{5})^n$, където $3 + 2\sqrt{5} \approx 7.472135954$. Важното е, че $3 + 2\sqrt{5} > 2$. Лесно се вижда, че за нехомогенната част $n^{\sqrt{n}}$ е вярно, че $n^{\sqrt{n}} \prec 2^n$: в сила е $n^{\sqrt{n}} = 2^{\lg\left(n^{\sqrt{n}}\right)} = 2^{\sqrt{n}\lg n}$, а очевидно $2^{\sqrt{n}\lg n} \prec 2^n$, понеже $\sqrt{n}\lg n \prec n$. Щом $n^{\sqrt{n}} \prec 2^n$ и $2^n \prec (3 + 2\sqrt{5})^n$, вярно е, че $n^{\sqrt{n}} \prec (3 + 2\sqrt{5})^n$. Близко е до ума, че в такъв случай асимптотиката се задава от големината на дървото на рекурсията, която е $\Theta((3 + 2\sqrt{5})^n)$. Да го докажем по индукция.

Първо да докажем, че $R(n) = O((3 + 2\sqrt{5})^n)$. Дали има константа $c > 0$, такава че $R(n) \leqslant c(3 + 2\sqrt{5})^n$ за всички достатъчно големи n? Индуктивните предположения са

$$R(n-1) \leqslant c(3 + 2\sqrt{5})^{n-1}$$
$$R(n-2) \leqslant c(3 + 2\sqrt{5})^{n-2}$$

Тогава имаме

$$R(n) \leqslant 6c(3 + 2\sqrt{5})^{n-1} + 11c(3 + 2\sqrt{5})^{n-2} + n^{\sqrt{n}}$$

Дали има $c > 0$, за което дясната страна е ограничена отгоре от $c(3 + 2\sqrt{5})^n$? За съжаление, не. Това неравенство

$$6c(3 + 2\sqrt{5})^{n-1} + 11c(3 + 2\sqrt{5})^{n-2} + n^{\sqrt{n}} \leqslant c(3 + 2\sqrt{5})^n$$

не е вярно, понеже

$$6c(3 + 2\sqrt{5})^{n-1} + 11c(3 + 2\sqrt{5})^{n-2} =$$
$$c\left(6(3 + 2\sqrt{5})^{n-1} + 11(3 + 2\sqrt{5})^{n-2}\right) =$$
$$c(3 + 2\sqrt{5})^n \left(\frac{6}{3 + 2\sqrt{5}} + \frac{11}{(3 + 2\sqrt{5})^2}\right) = c(3 + 2\sqrt{5})^n$$

Убедете се сами, че $\frac{6}{3+2\sqrt{5}} + \frac{11}{(3+2\sqrt{5})^2} = 1$. Ще засилим твърдението – ще извадим една експоненциална функция от $(3+2\sqrt{5})^{n-1}$, но с основа, по-малка от $3+2\sqrt{5}$. Сега доказваме, че съществува константа $c > 0$, такава че $R(n) \leqslant c(3+2\sqrt{5})^n - 2^n$ за всички достатъчно големи n. Забележете, че $c(3+2\sqrt{5})^n - 2^n$ има само положителни стойности за всички достатъчно големи n, независимо от това колко малка е $c > 0$. Индуктивните предположения са

$$R(n-1) \leqslant c(3+2\sqrt{5})^{n-1} - 2^{n-1}$$
$$R(n-2) \leqslant c(3+2\sqrt{5})^{n-2} - 2^{n-2}$$

Тогава имаме

$$R(n) \leqslant 6c(3+2\sqrt{5})^{n-1} - 6\cdot 2^{n-1} + 11c(3+2\sqrt{5})^{n-2} - 11\cdot 2^{n-2} + n^{\sqrt{n}}$$

Преобразуваме дясната страна

$$6c(3+2\sqrt{5})^{n-1} - 6\cdot 2^{n-1} + 11c(3+2\sqrt{5})^{n-2} - 11\cdot 2^{n-2} + n^{\sqrt{n}} =$$
$$c(3+2\sqrt{5})^n \left(\frac{6}{3+2\sqrt{5}} + \frac{11}{(3+2\sqrt{5})^2}\right) - 6\cdot 2^{n-1} - 11\cdot 2^{n-2} + n^{\sqrt{n}} =$$
$$c(3+2\sqrt{5})^n - 6\frac{2^n}{2} - 11\frac{2^n}{4} + n^{\sqrt{n}} =$$
$$c(3+2\sqrt{5})^n - \frac{23}{4}2^n + n^{\sqrt{n}}$$

Дали има $c > 0$, такава че

$$c(3+2\sqrt{5})^n - \frac{23}{4}2^n + n^{\sqrt{n}} \leqslant c(3+2\sqrt{5})^n - 2^n \leftrightarrow$$
$$-\frac{19}{4}2^n + n^{\sqrt{n}} \leqslant 0$$

за всички достатъчно големи n? Със сигурност да, понеже лявата страна е отрицателна за всички достатъчно големи n: както вече показахме, $2^n > n^{\sqrt{n}}$.

Сега ще докажем, че $R(n) = \Omega((3+2\sqrt{5})^n)$. Дали има константа $d > 0$, такава че $R(n) \geqslant d(3+2\sqrt{5})^n$ за всички достатъчно големи n? Индуктивните предположения са

$$R(n-1) \geqslant d(3+2\sqrt{5})^{n-1}$$
$$R(n-2) \geqslant d(3+2\sqrt{5})^{n-2}$$

Тогава имаме

$$R(n) \geqslant 6d(3+2\sqrt{5})^{n-1} + 11d(3+2\sqrt{5})^{n-2} + n^{\sqrt{n}}$$

Дали има $d > 0$, за което дясната страна е ограничена отдолу от $d(3+2\sqrt{5})^n$?

$$6d(3+2\sqrt{5})^{n-1} + 11d(3+2\sqrt{5})^{n-2} + n^{\sqrt{n}} \geqslant d(3+2\sqrt{5})^n \leftrightarrow$$
$$d(3+2\sqrt{5})^n \left(\frac{6}{3+2\sqrt{5}} + \frac{11}{(3+2\sqrt{5})^2}\right) + n^{\sqrt{n}} \geqslant d(3+2\sqrt{5})^n \leftrightarrow$$
$$d(3+2\sqrt{5})^n + n^{\sqrt{n}} \geqslant d(3+2\sqrt{5})^n \leftrightarrow n^{\sqrt{n}} \geqslant 0$$

Да, със сигурност има такава, да кажем $d = 1$.

Решаваме $S(n) = 4S\left(\frac{n}{2}\right) + n^2(\lg n)^3$. Ще го решим с разширението на МТ. $a = 4$, $b = 2$, $f(n) = n^2(\lg n)^3$. Тогава $\log_b a = \log_2 4 = 2$. Разширението на МТ е приложимо. Съгласно него, $S(n) \asymp n^2(\lg n)^4$.

Решаваме $T(n) = 4T\left(\frac{n}{2}\right) + n^{\lg n}$. Ще го решим с МТ. $a = 4$, $b = 2$, $f(n) = n^{\lg n}$. Тогава $\log_b a = \log_2 4 = 2$. Сравняваме $f(n)$ с $n^{\log_b a}$. Веднага се вижда, че $f(n) \geq n^{\log_b a}.n^{\epsilon}$ за каквато искаме положителна константа ϵ. Ще опитаме да използваме третия случай на МТ. Да видим дали условието за регулрност е в сила. Тоест, дали съществува константа c, такава че $0 < c < 1$ и за всички достатъчно големи n е в сила $a \cdot f\left(\frac{n}{b}\right) \leqslant c \cdot f(n)$.

$$4\left(\frac{n}{2}\right)^{\lg \frac{n}{2}} \leqslant cn^{\lg n} \leftrightarrow 4\frac{n^{\lg \frac{n}{2}}}{2^{\lg \frac{n}{2}}} \leqslant cn^{\lg n} \leftrightarrow 4\frac{n^{(\lg n)-1}}{2^{(\lg n)-1}} \leqslant cn^{\lg n} \leftrightarrow 4\frac{\frac{n^{\lg n}}{n}}{\frac{2^{\lg n}}{2}} \leqslant cn^{\lg n} \leftrightarrow$$

$$4\frac{\frac{n^{\lg n}}{n}}{\frac{n}{2}} \leqslant cn^{\lg n} \leftrightarrow 8\frac{n^{\lg n}}{n^2} \leqslant cn^{\lg n} \leftrightarrow 8 \leqslant cn^2$$

Всяка константа c, такава че $0 < c < 1$, върши работа. Условието за регулярност е изпълнено и третият случай на МТ е приложим. Съгласно него, $T(n) \asymp n^{\lg n}$.

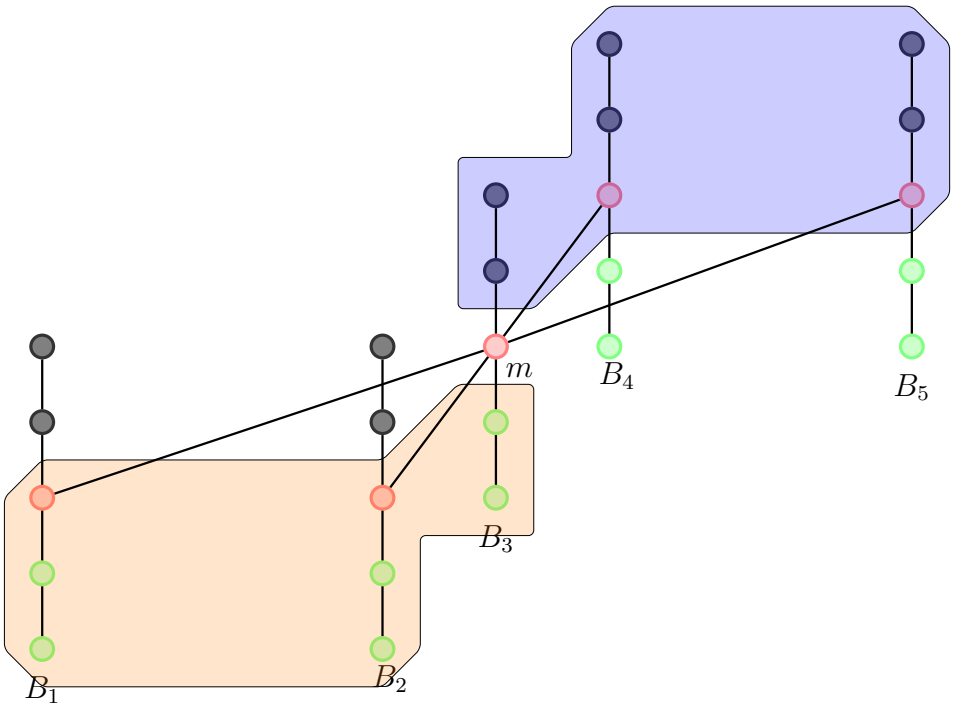
Задача 3: Припомнете си алгоритъма SELECT$(A[1, \ldots, n], k)$, изучаван на лекции. В тази задача се има предвид псевдокодът на този алгоритъм точно както е в лекционните записки. Нека $n = 25$ и нека във входа $A[1, \ldots, 25]$ се намира някоя пермутация на $\{1, 2, \ldots, 25\}$, но не непременно същата като тази в примера от лекции.

1 т. Първо, кажете коя е медианата на $\{1, 2, \ldots, 25\}$.

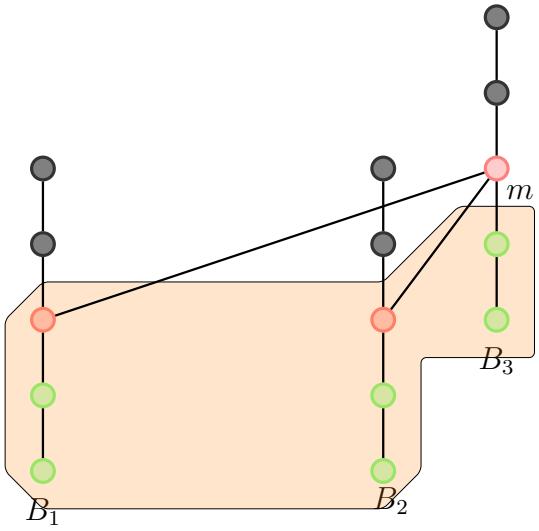
15 т. Второ, намерете такъв вход-пермутация на $\{1, 2, \ldots, 25\}$, че върху него алгоритъмът присвоява на m (на ред 11) стойност, която е максимално отдалечена от медианата на A. Каква стойност за m намерихте? Обосновете добре отговора си.

Решение: Медианата на $\{1, 2, \ldots, 25\}$ е 13.

Иска се да намерим вход, за който медианата на медианите m е максимално далече от 13. Да си припомним диаграмата на Hasse на данните след m. Сега $n = 25$, $q = 5$ и диаграмата изглежда така:



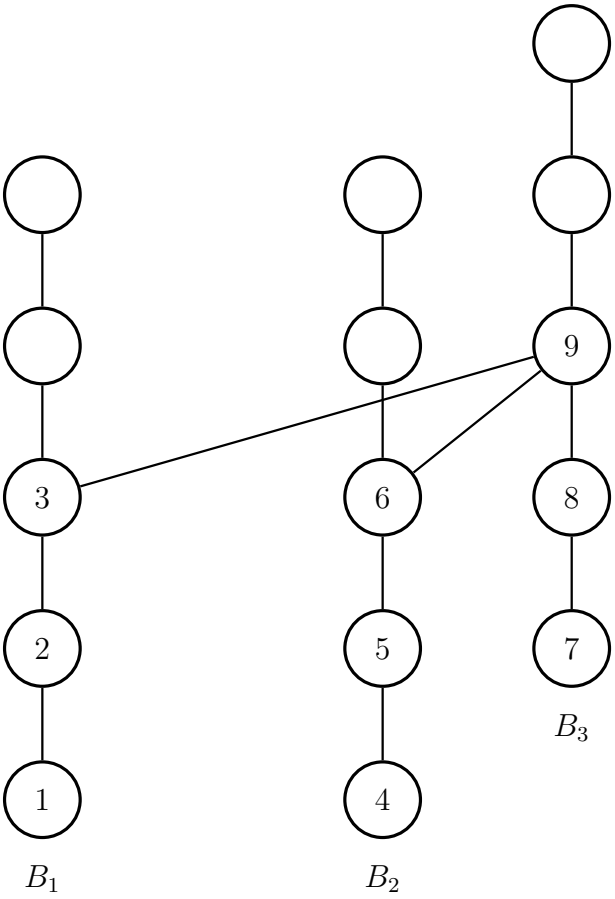
БОО, да се фокусираме върху елементите, по-малки от m:



m е задължително по-голямо число от трите най-малки елемента на B_1, от трите най-малки елемента на B_2, и от двата по-малки от него елемента на B_3, но само толкова. Възможно е всеки друг елемент да е по-голям от m. Така че точна долна граница за броят на елементите, по-малки от m, е $3+3+2=8$. Това може да се изведе и чрез формулата от лекционните записки

$$3\left\lfloor\frac{q-1}{2}\right\rfloor+2=3\left\lfloor\frac{5-1}{2}\right\rfloor+2=3\cdot 2+2=8$$

Тази ситуация е постижима, ако $m=9$. Въпросните осем елемента са числата $1,\ldots,8$. Ето едно тяхно възможно разположение:



За да се окажат по този начин числата 1, . . . , 9 след сортирането на $B_1, \ldots, B_5$ и намирането на медианата от медианите, достатъчно е

- 1, 2 и 3 да са в B_1 поначало,
- 4, 5 и 6 да са в B_2 поначало,
- 7, 8 и 9 да са в B_3 поначало.

Всички други числа са по-големи от тези и наистина няма значение кое от тях (по-големите от 9) в кой масив B_i се намира.

За да се получи желанато "разпределение" на 1, . . . , 9 в на B_1, B_2 и B_3 преди сортирането на масивите B_i, достатъчно е във входния A, числата 1, 2, 3 да са в първата петица, 5, 6, 7 да са във втората петица, а 7, 8, 9 да са в третата петица. С други думи, A от входа да е

$$A = [1, 2, 3, *, *, 5, 6, 7, *, *, 7, 8, 9, *, *, *, *, *, *, *, *, *, *, *, *, *]$$

"$*$" ознавача "кое да е число, по-голямо от 9".