---

**Задача 1:** Нека $k \in \mathbb{N}^+$, $b_1, \ldots, b_k \in \mathbb{R}$, $b_1, \ldots, b_k > 1$ и $\frac{1}{b_1} + \cdots + \frac{1}{b_k} = 1$. Докажете по индукция, че рекурентното уравнение

$$T(n) = T\left(\frac{n}{b_1}\right) + \cdots + T\left(\frac{n}{b_k}\right) + n$$

има решение $T(n) = \Theta(n \lg n)$.

**Решение:** ❶ Първо ще докажем, че $T(n) = O(n \lg n)$. По дефиниция, това е същото като

$$\exists c > 0 \; \forall n \nearrow : T(n) \leq cn \lg n \tag{1}$$

Ползваме силна индукция. Индуктивното предположение е, че за $\ell \in \{a, \ldots, n-1\}$ : $T(\ell) \leq c\ell \lg \ell$, където $a$ е някаква константа, чиято стойност зависи от имплицитната база. В частност:

$$T\left(\frac{n}{b_1}\right) \leq c\frac{n}{b_1} \lg \frac{n}{b_1}$$

$$\ldots$$

$$T\left(\frac{n}{b_k}\right) \leq c\frac{n}{b_k} \lg \frac{n}{b_k}$$

От дефиницията на $T(n)$ и индуктивното предположение заключаваме, че

$$T(n) \leq \frac{cn}{b_1} \lg \frac{n}{b_1} + \cdots + \frac{cn}{b_k} \lg \frac{n}{b_k} + n \tag{2}$$

Разглеждаме (2):

$$\begin{aligned}
T(n) &\leq \left(\frac{cn}{b_1} \lg \frac{n}{b_1} + \cdots + \frac{cn}{b_k} \lg \frac{n}{b_k}\right) + n \\
&= \left(\frac{cn}{b_1}(\lg n - \lg b_1) + \cdots + \frac{cn}{b_k}(\lg n - \lg b_k)\right) + n \\
&= \left(\left(\frac{cn \lg n}{b_1} - \frac{cn \lg b_1}{b_1}\right) + \cdots + \left(\frac{cn \lg n}{b_k} - \frac{cn \lg b_k}{b_k}\right)\right) + n \\
&= \left(\frac{cn \lg n}{b_1} + \cdots + \frac{cn \lg n}{b_k}\right) - \left(\frac{cn \lg b_1}{b_1} + \cdots + \frac{cn \lg b_k}{b_k}\right) + n \\
&= cn \lg n \underbrace{\left(\frac{1}{b_1} + \cdots + \frac{1}{b_k}\right)}_{1} - cn\left(\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k}\right) + n \\
&= cn \lg n - cn\left(\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k}\right) + n
\end{aligned}$$

Нека $\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k} = B$. Тогава

$$T(n) \leq cn \lg n - cnB + n$$

Твърдим, че съществува положително $c$, такова че

$$cn \lg n - cnB + n \leq cn \lg n$$

Наистина,

$$cn \lg n - cnB + n \leq cn \lg n \;\leftrightarrow\; -cnB + n \leq 0 \;\leftrightarrow\; 1 \leq cB \;\leftrightarrow\; c \geq \frac{1}{B}$$

Такова $c$ съществува, понеже $B > 0$, тоест, $\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k} > 0$. А това е вярно, понеже $b_i > 1$ за $1 \leq i \leq k$ по условие. Доказахме (1). ✓

❷ Сега ще докажем, че $T(n) = \Omega(n \lg n)$. По дефиниция, това е същото като

$$\exists d > 0 \; \forall n \nearrow : T(n) \geq dn \lg n \tag{3}$$

Ползваме силна индукция. Индуктивното предположение е, че за $\ell \in \{e, \ldots, n-1\}$ : $T(\ell) \geq d\ell \lg \ell$, където $e$ е някаква константа, чиято стойност зависи от имплицитната база. В частност:

$$T\left(\frac{n}{b_1}\right) \geq d\frac{n}{b_1} \lg \frac{n}{b_1}$$

$$\cdots$$

$$T\left(\frac{n}{b_k}\right) \geq d\frac{n}{b_k} \lg \frac{n}{b_k}$$

От дефиницията на $T(n)$ и индуктивното предположение заключаваме, че

$$T(n) \geq \frac{dn}{b_1} \lg \frac{n}{b_1} + \cdots + \frac{dn}{b_k} \lg \frac{n}{b_k} + n \tag{4}$$

Разглеждаме (4):

$$\begin{aligned}
T(n) &\geq \frac{dn}{b_1} \lg \frac{n}{b_1} + \cdots + \frac{dn}{b_k} \lg \frac{n}{b_k} + n \\
&= \left(\frac{dn}{b_1}(\lg n - \lg b_1) + \cdots + \frac{dn}{b_k}(\lg n - \lg b_k)\right) + n \\
&= \left(\left(\frac{dn \lg n}{b_1} - \frac{dn \lg b_1}{b_1}\right) + \cdots + \left(\frac{dn \lg n}{b_k} - \frac{dn \lg b_k}{b_k}\right)\right) + n \\
&= \left(\frac{dn \lg n}{b_1} + \cdots + \frac{dn \lg n}{b_k}\right) - \left(\frac{dn \lg b_1}{b_1} + \cdots + \frac{dn \lg b_k}{b_k}\right) + n \\
&= dn \lg n \underbrace{\left(\frac{1}{b_1} + \cdots + \frac{1}{b_k}\right)}_{1} - dn \left(\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k}\right) + n \\
&= dn \lg n - dn \left(\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k}\right) + n
\end{aligned}$$

Нека $\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k} = B$. Тогава

$$T(n) \geq dn \lg n - dnB + n$$

Твърдим, че съществува положително $d$, такова че

$$dn \lg n - dnB + n \geq dn \lg n$$

Наистина,

$$dn \lg n - dnB + n \geq dn \lg n \;\leftrightarrow\; -dnB + n \geq 0 \;\leftrightarrow\; 1 \geq dB \;\leftrightarrow\; d \leq \frac{1}{B}$$

Такова $d$ съществува, понеже $B > 0$, тоест, $\frac{\lg b_1}{b_1} + \cdots + \frac{\lg b_k}{b_k} > 0$. А това е вярно, понеже $b_i > 1$ за $1 \leq i \leq k$ по условие. Доказахме (3). ✓

**Задача 2:** (ас. Мартин Георгиев) Един масив $A[1\,..\,n]$ ще наричаме *$p$-стръмен*, ако съществува индекс $i$ такъв, че за всеки индекс $j$ е изпълнено $A[j] + |i - j| \geq p$. Предложете ефикасен алгоритъм, който при подаден масив $A[1\,..\,n]$ намира максималното $p$, за което $A$ е $p$-стръмен. Докажете, че алгоритъмът ви работи коректно и анализирайте сложността му по време.

**Решение:** (ас. Мартин Георгиев) Задачата може да се опрости. За целта въвеждаме следните две дефиниции.

**Определение 1** *Масив $A[1\,..\,n]$ е* $p$-нарастващ по отношение на $i$, *ако за всяко $j$, такова че $1 \leq j \leq i$, е в сила $A[j] + i - j \geq p$.*

**Определение 2** *Масив $A[1\,..\,n]$ е* $p$-намаляващ по отношение на $i$, *ако за всяко $j$, такова че $i \leq j \leq n$, е в сила $A[j] + j - i \geq p$.*

Извършваме следното наблюдение.

> За произволен масив $A[1\,..\,n]$ и за произволно $p$, $A$ е $p$-стръмен тогава и само тогава, когато съществува индекс $i$, за който $A$ е $p$-нарастващ по отношение на $i$ и $p$-намаляващ по отношение на $i$.

Така, за да определим максималното $p$, за което $A$ е $p$-стръмен, е достатъчно да изберем най-голямата от най-големите общи стойности на нарастване и намаляване относно всеки индекс от 1 до $n$. Следният алгоритъм реализира тази идея:

```
MAX_BUMP(A[1..n])
1  B ← MAX_INCREASING(A)
2  C ← MAX_DECREASING(A)
3  p ← −∞
4  for i ← 1 to n
5      p ← max(p, min(B[i], C[i]))
6  return p
```

Тук функциите MAX_INCREASING и MAX_DECREASING връщат масиви, чиито елементи отговарят съответно на максималните стойности на нарастване и намаляване на $A$ относно съответните индекси (с други думи за всеки индекс $i$ между 1 и $n$, $B[i]$ съдържа най-голямата стойност на нарастване на $A$ относно $i$, а $C[i]$ съдържа най-голямата стойност на намаляване на $A$ относно $i$). Следният алгоритъм реализира MAX_INCREASING за линейно време:

```
MAX_INCREASING(A[1..n]: масив от цели числа)
1  създай масив B[0..n]
2  B[0] ← ∞
3  for i ← 1 to n
4      B[i] ← min(B[i − 1] + 1, A[i])
5  return B[1..n]
```

Алгоритъмът последователно запълва максималните стойности на нарастване на $A$ по отношение на индексите от 1 до $n$, като коректността му е следствие от наблюдението, че $A$ е $p$-нарастващ по отношение на $i$ точно тогава, когато $A[i] \geq p$ и $A$ е $(p-1)$-нарастващ по отношение на $i-1$. По-формално, инвариант на цикъла на MAX_INCREASING ще бъде:

**Инвариант.** При всяко достигане на ред 3 за всеки индекс $j$ между $\mathbf{0}$ и $i-1$, $B[j]$ държи максималната стойност на нарастване на $A$ по отношение на $j$.

Доказателството на инварианта е по индукция по броя на достиганията на ред 3.

**База.** Да разгледаме първото достигане на ред 3, когато $i$ е 1. Тогава интервалът $[0\,..\,i-1] = [0\,..\,0]$ има единствен елемент 0. Тъй като максималната стойност на нарастване на $A$ по отношение на 0 е $\infty = B[0]$, то твърдението е изпълнено. ✓

**Поддръжка.** Да допуснем, че твърдението е вярно при някое достигане на ред 3, което не е последно. Ще покажем, че $B[i]$ бива остойностено с максималната стойност на нарастване на $A$ относно $i$. Разглеждаме два случая:

1 сл. $A[i] \geq B[i-1]+1$. Тогава $B[i]$ бива остойностено с $B[i-1]+1$ и това остойностяване е коректно:

- $A$ е $(B[i-1]+1)$-нарастващ в $i$. За да докажем това трябва да покажем, че за всяко $j$ между 1 и $i$ е изпълнено, че $A[j]+i-j \geq B[i-1]+1$. Е, ако $j=i$, то $A[j]+i-j = A[i] \geq B[i-1]+1$, а ако пък $j<i$, то по допускане ще е изпълнено

  $$A[j]+i-j = (A[j]+(i-1)-j)+1 \geq B[i-1]+1$$

  защото $A$ е $(B[i-1])$-нарастващ в $i-1$.

- $A$ не е $p$-нарастващ относно $i$ за никое $p > B[i-1]+1$. Да допуснем противното. Нека $A$ е $p$-нарастващ относно $i$ за някое $p > B[i-1]+1$. Така, по дефиниция, за всяко $j \leq i$ ще бъде изпълнено

  $$A[j]+i-j \geq p$$

  което може да се запише и като

  $$A[i]+(i-1)-j \geq p-1$$

  Последното неравенство влече, че $A$ е $(p-1)$-нарастващ в $i-1$. По допускане обаче $B[i-1]$ е масималната стойност на нарастване на $A$ относно $i-1$, а $p-1 > B[i-1]$. Така достигнахме до противоречие, откъдето допускането ни се оказва грешно и $A$ наистина не е $p$-нарастващ относно $i$ за никое $p > B[i-1]+1$.

2 сл. $A[i] < B[i-1]+1$. Тогава $B[i]$ бива остойностено с $A[i]$ и това остойностяване е коректно:

- $A$ е $A[i]$-нарастващ в $i$. За да докажем това трябва да покажем, че за всяко $j$ между 1 и $i$ е изпълнено, че $A[j] + i - j \geq A[i]$. Е, ако $j = i$, то $A[j] + i - j = A[i] \geq A[i]$, а ако пък $j < i$, то по допускане ще е изпълнено

  $$A[j] + i - j = (A[j] + (i-1) - j) + 1 \geq B[i-1] + 1 > A[i]$$

  защото $A$ е $(B[i-1])$-нарастващ в $i-1$.

- $A$ не е $p$-нарастващ относно $i$ за никое $p > A[i]$. За да докажем това, трябва да покажем, че съществува индекс $j$ между 1 и $i$, за който $A[j] + i - j < p$. Е, ако вземем $j = i$, то това ще е изпълнено - $A[j] + i - j = A[i] < p$.

Тъй като и в двата случая $B[i]$ бива остойностено с максималната стойност на нарастване на $A$ относно $i$, при следващото достигане на ред 3, в термините на новото $i' = i + 1$, масивът $B[0\,..\,i'-1] = B[0\,..\,i]$ ще съдържа максималните стойности на нарастване на $A$ по отношение на съответните индекси. ✓

**Терминация.** При последното достигане на ред 3, стойността на $i$ ще бъде $n+1$ и, като следствие от инварианта, масивът $B[1\,..\,n]$ ще държи максималните стойности на нарастване на $A$ по отношение на съответние индекси. Така на ред 5 алгоритъмът коректно връща $B[1\,..\,n]$.

MAX_DECREASING може да се реализира аналогично, като посоката на обхождане на $A$ ще бъде отзад напред. Алтернативно можем да се възползваме от наблюдението, че за произволен масив $A[1\,..\,n]$, $A$ е $p$-нарастващ по отношение на $i$ тогава и само тогава, когато reverse$(A)$ е $p$-намаляващ по отношение на $n-i+1$. Така една по-сбита реализация на MAX_DECREASING може да бъде:

```
MAX_DECREASING(A[1..n]: масив от цели числа)
  1  return reverse(MAX_INCREASING(reverse(A)))
```

Примерните реализации на MAX_INCREASING и MAX_DECREASING са линейни по време и по памет. Така сложността по време и по памет на MAX_BUMP също ще бъде линейна.

**Задача 3:** *Кръгов масив* е всеки масив $A[1 \dots n]$, такъв че $n \geq 3$ и освен това, $A[1]$ и $A[n]$ са съседи в масива. В контекста на кръговите масиви, разстоянието между две различни позиции в масива се дефинира така:

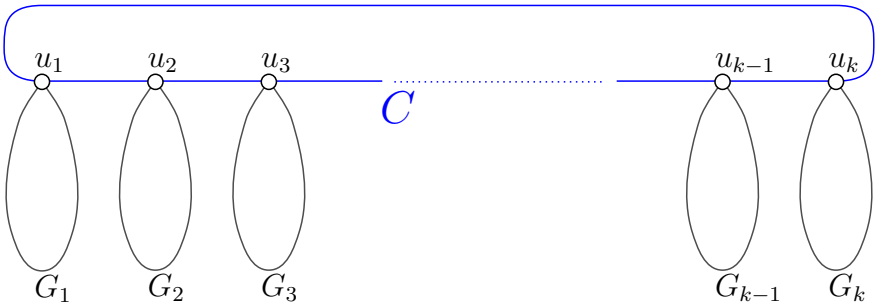
$$\forall i, j \in \{1, 2, \dots, n\}, i \neq j : \text{dist}(i, j) = \max\{|i - j|, n - |i - j|\}$$

Конструирайте линеен алгоритъм, който получава като вход кръгов масив $A[1 \dots n]$ и който намира

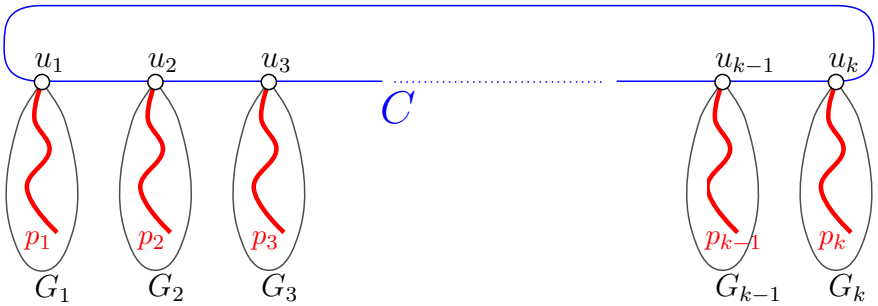
$$\max\{A[i] + A[j] + \text{dist}(i, j) \mid i, j \in \{1, 2, \dots, n\}, i \neq j\}$$

Докажете коректността на Вашия алгоритъм и анализирайте сложността му по време.

**Решение:** Задачата има графова интерпретация. Тъй като $n$ в контекста на графите по подразбиране има смисъл на броя на върховете, ползваме $k$ вместо $n$. Да си представим свързан граф $G$, в който има цикъл $C = u_1, u_2, \dots, u_k, u_1$, такъв че, ако изтрием от $G$ ребрата на $C$, то $G$ ще се разпадне на $k$ свързани компоненти $G_1$, …, $G_k$, като $u_i \in G_i$ за $i \in \{1, \dots, k\}$. Ето рисунка на $G$, като ребрата на $C$ са в синьо.




Да допуснем, че за $i \in \{1, \dots, k\}$, $p_i$ е най-дълъг път в $G_i$ с един край $u_i$. Очевидно $|p_i| \geq 0$, като границата е точна, понеже $G_i$ може да е тривиален.

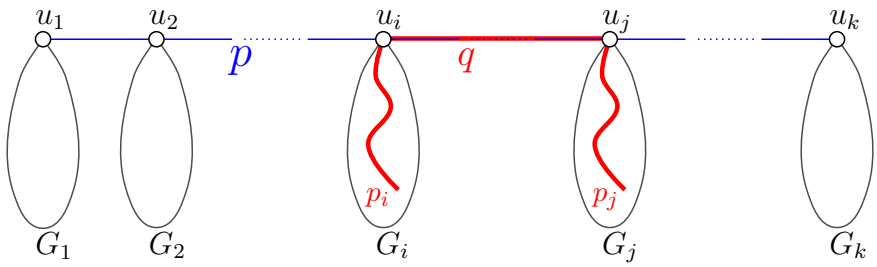



Дадени са дължините $|p_i|$, за $i \in \{1, \dots, k\}$. Трябва да се намери дължината на най-дълъг път $q$ в $G$, който съдържа поне едно ребро от $C$.

Да фиксираме произволно ребро от $C$. БОО, нека това е реброто $(u_1, u_k)$. Очевидно $q$ или съдържа $(u_1, u_k)$, или не съдържа $(u_1, u_k)$.

- Ако $q$ не съдържа $(u_1, u_k)$, можем да мислим, че $(u_1, u_k)$ е изтрито от графа. Тогава $C$ се превръща в път, който ще наречем $p$. Тогава $q$ се състои от "слепването" на тези три пътя:
  - $p_i$,
  - подпътя на $p$ между $u_i$ и $u_j$,
  - и $p_j$,

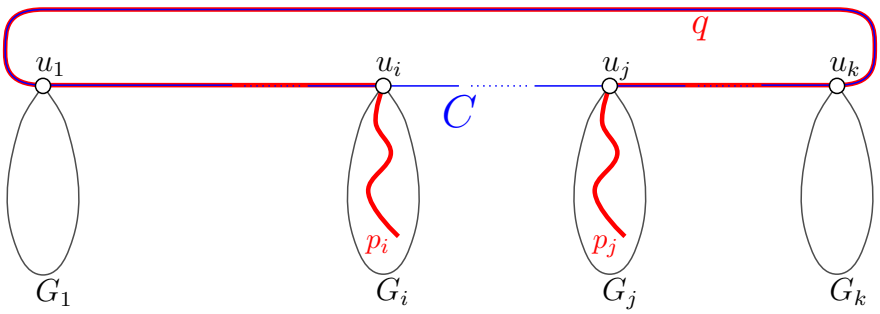
  за някакви $i$ и $j$, такива че $1 \leq i < j \leq k$.




  При това положение, $|q| = |p_i| + j - i + |p_j|$.

- Ако $q$ съдържа $(u_1, u_k)$, то $q$ се състои от "слепването" на тези пет пътя:
  - $p_i$,
  - подпътя на $C - (u_1, u_k)$ между $u_1$ и $u_i$,
  - реброто $(u_1, u_k)$,
  - подпътя на $C - (u_1, u_k)$ между $u_j$ и $u_k$,
  - и $p_j$,

  за някакви $i$ и $j$, такива че $1 \leq i < j \leq k$.




  При това положение, $|q| = |p_i| + (i-1) + 1 + (k-j) + |p_j| = |p_i| + k - (j-i) + |p_j|$.

Да забравим за графовата интерпретация и да се фокусираме върху дадената задача. Следният алгоритъм я решава в линейно време.

AlgCircularDist($A[1 \dots n]$: кръгов масив от естествени числа)

```
1   нека B[0...n] и C[1...n] са обикновени (линейни) масиви от ест. числа
2   B[0] ← 0
3   for i ← 1 to n
4       B[i] ← max{B[i − 1], A[i] − (i − 1)}
5       C[i] ← B[i − 1] + A[i] + (i − 1)
6   x ← max{C[i] | 1 ≤ i ≤ n}
7   for i ← 1 to n
8       B[i] ← max{B[i − 1], A[i] + (i − 1)}
9   C[n] ← A[n] + 1
10  for i ← n − 1 downto 2
11      C[i] ← max{C[i + 1], A[i] + n − (i − 1)}
12  y ← max{B[i] + C[i + 1] | 1 ≤ i ≤ n − 1}
13  return max{x, y}
```

На лекции сме изучавали алгоритъма Alg Max Sum Plus Dist (Алгоритъм 3 в лекционните записки), който решава "половината задача", а именно случая, в който масивът $A[1 \dots n]$ се третира като линеен масив. Кодът на редове 1–6 е практически същият като Alg Max Sum Plus Dist. Следователно, съгласно изучаваното на лекции, след присояването на ред 6 е вярно, че

$$x = \max\{A[i] + A[j] + j - i \mid 1 \le i < j \le n\} \tag{5}$$

Да решим другата "половина от задачата", в която масивът $A[1 \dots n]$ се третира като кръгов, но, ако мислим за графовата интерпретация, се търси максимална дължина на път, който съдържа реброто $(A[1], A[n])$. Забелязваме, че първият **for**-цикъл на редове 3–5 не променя елемента $B[0]$, така че $B[0] = 0$ при първото достигане на ред 7.

**Лема 1** *След терминирането на втория* ***for****-цикъл на редове 7–8:*

$$\forall i \in \{1, \dots, n\} : B[i] = \max\{A[1], A[2] + 1, A[3] + 2, \dots, A[i] + (i - 1)\} \tag{6}$$

**Доказателство:** Инвариант на цикъла е този. При всяко достигане на ред 7,

$$\forall j \in \{1, \dots, i\} : B[j - 1] = \max\{A[1], A[2] + 1, A[3] + 2, \dots, A[j - 1] + (j - 2)\} \tag{7}$$

Базата е първото достигане на ред 7. Тогава $i = 1$. Заместваме $i$ с 1 в (7) и получаваме

$$\forall j \in \{1, \dots, 1\} : B[j - 1] = \max\{A[1], A[2] + 1, A[3] + 2, \dots, A[j - 1] + (j - 2)\}$$

което е същото като $B[0] = \max\{A[1], \dots, A[0] + (-1)\}$. Забележете, че това е същото като $B[0] = \max \emptyset$, понеже наредено множество от индексите $\{1, \dots, 0\}$ е празното множество. В случая $\max \emptyset = 0$ (а не $-\infty$), защото разглеждаме само естествени числа. И наистина, $B[0] = 0$ в този момент, както вече отбелязахме. Доказахме базовия случай. ✓

Да допуснем, че (7) е в сила при някое достигане на ред 7, което не е последното. Подробно написано, допускането е

$$\begin{aligned}
B[0] &= \max\{A[1], \ldots, A[0] + (-1)\} \quad // \text{ дясната страна е } \max\emptyset \\
B[1] &= \max\{A[1]\} \\
B[2] &= \max\{A[1], A[2] + 1\} \\
B[3] &= \max\{A[1], A[2] + 1, A[3] + 2\} \\
&\ldots \\
B[i-1] &= \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[i-1] + (i-2)\}
\end{aligned}$$

След присвояването на ред 8, вече е вярно, че

$$\begin{aligned}
B[0] &= \max\{A[1], \ldots, A[0] + (-1)\} \quad // \text{ дясната страна е } \max\emptyset \\
B[1] &= \max\{A[1]\} \\
B[2] &= \max\{A[1], A[2] + 1\} \\
B[3] &= \max\{A[1], A[2] + 1, A[3] + 2\} \\
&\ldots \\
B[i-1] &= \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[i-1] + (i-2)\} \\
B[i] &= \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[i-1] + (i-2), A[i] + (i-1)\}
\end{aligned}$$

Изпълнението отива на ред 7, където $i$ се инкрементира. Спрямо новото $i$ отново е вярно, че

$$\forall j \in \{1, \ldots, i\} : B[j-1] = \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[j-1] + (j-2)\}$$

Във фазата **Терминация** на доказателството, разлеждаме последното достигане на ред 7. Тогава $i = n + 1$. Заместваме $i$ с $n + 1$ в (7) и получаваме

$$\forall j \in \{1, \ldots, n+1\} : B[j-1] = \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[j-1] + (j-2)\}$$

Тогава

$$\forall j \in \{2, \ldots, n+1\} : B[j-1] = \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[j-1] + (j-2)\}$$

Сменяйки променливата, правим $j - 1 = i$ и получаваме

$$\forall i \in \{1, \ldots, n\} : B[i] = \max\{A[1], A[2] + 1, A[3] + 2, \ldots, A[i] + (i-1)\}$$

Което и трябваше да докажем. □

**Лема 2** *След терминирането на третия* ***for****-цикъл на редове 10–11:*

$$\forall i \in \{2, \ldots, n\} : C[i] = \max\{A[k] + n - (k-1) \mid k \in \{i, \ldots, n\}\} \tag{8}$$

**Доказателство:** Инвариант на цикъла е този. При всяко достигане на ред 10,

$$\forall j \in \{i+1, \ldots, n\} : C[j] = \max\{A[k] + n - (k-1) \mid k \in \{j, \ldots, n\}\} \tag{9}$$

Базата е първото достигане на ред 10. Тогава $i = n-1$. Заместваме $i$ с $n-1$ в (9) и получаваме

$$\forall j \in \{n, \ldots, n\} : C[j] = \max\{A[k] + n - (k-1) \mid k \in \{j, \ldots, n\}\}$$

което е същото като

$$C[n] = \max\{A[k] + n - (k-1) \mid k \in \{n, \ldots, n\}\}$$

което е същото като

$$C[n] = A[n] + n - (n-1)$$

Но на ред 9, $C[n]$ получава точно тази стойност, ако представим $+1$ като $n-(n-1)$. ✓

Да допуснем, че (9) е в сила при някое достигане на ред 10, което не е последното. Подробно написано, допускането е

$$\begin{aligned}
C[n] &= A[n] + 1 \\
C[n-1] &= \max\{A[n] + 1, A[n-1] + 2\} \\
C[n-2] &= \max\{A[n] + 1, A[n-1] + 2, A[n-2] + 3\} \\
&\ldots \\
C[i+1] &= \max\{A[n] + 1, A[n-1] + 2, \ldots, A[i+1] + n - i\}
\end{aligned}$$

След това на ред 11 на $C[i]$ се присвоява $\max\{C[i+1], A[i] + n - (i-1)\}$. В сила е

$$\begin{aligned}
C[n] &= A[n] + 1 \\
C[n-1] &= \max\{A[n] + 1, A[n-1] + 2\} \\
C[n-2] &= \max\{A[n] + 1, A[n-1] + 2, A[n-2] + 3\} \\
&\ldots \\
C[i+1] &= \max\{A[n] + 1, A[n-1] + 2, \ldots, A[i+1] + n - i\} \\
C[i] &= \max\{A[n] + 1, A[n-1] + 2, \ldots, A[i+1] + n - i, A[i] + n - (i-1)\}
\end{aligned}$$

Изпълнението отива отново на ред 10, където $i$ се декрементира. По отношение на новото $i$ имаме право да кажем, че

$$\forall j \in \{i+1, \ldots, n\} : C[j] = \max\{A[k] + n - (k-1) \mid k \in \{j, \ldots, n\}\}$$

И така, (9) остава в сила.

При последното достигане на ред 10, $i$ съдържа 1. Заместваме в инварианта $i$ с 1 и получаваме

$$\forall j \in \{2, \ldots, n\} : C[j] = \max\{A[k] + n - (k-1) \mid k \in \{j, \ldots, n\}\}$$

Но това е точно (8). □

**Лема 3** *След присвояването на ред 12, в сила е*

$$y = \max\{A[i] + A[j] + n - (j - i) \mid 1 \leq i < j \leq n\} \tag{10}$$

**Доказателство:** Следва веднага от Лема 1 и Лема 2. □

**Теорема 1** *На ред 13, алгоритъмът* AlgCircularDist($A[1 \ldots n]$) *връща*

$$\max\{A[i] + A[j] + dist(i, j) \mid i, j \in \{1, 2, \ldots, n\}, i \neq j\}$$

**Доказателство:** Както знаем от (5) и (10), при достигането на ред 13 са в сила

$$x = \max\{A[i] + A[j] + j - i \mid 1 \leq i < j \leq n\}$$
$$y = \max\{A[i] + A[j] + n - (j - i) \mid 1 \leq i < j \leq n\}$$

Съгласно дефиницията на $\mathrm{dist}(i, j)$, максимумът от тези двете е именно

$$\max\{A[i] + A[j] + \mathrm{dist}(i, j) \mid i, j \in \{1, 2, \ldots, n\}, i \neq j\}$$ □