Algoritmizace

Cvičení 00

Vladimír Kunc (kuncvlad)
FEL ČVUT

4. října 2018

Organizace

Všechny informace dostupné na
`https://cw.fel.cvut.cz/wiki/courses/b4b33alg`

- Cvičení
  - procvičování úloh
  - docházka, max. 2 absence
- Domácí úlohy
  - 8 úloh, 16 bodů
  - hodnocení úloh na testovací sadě dat
    - 10/10 — 2 body
    - 9/10 — 1 bod
    - $\leq$8/10 — 0 bodů

Příklady

Zadání příkladů

https://cw.fel.cvut.cz/wiki/_media/courses/a4b33alg/cviceni01upd.pdf

Hodí se znát

https://en.wikipedia.org/wiki/Arithmetic_progression

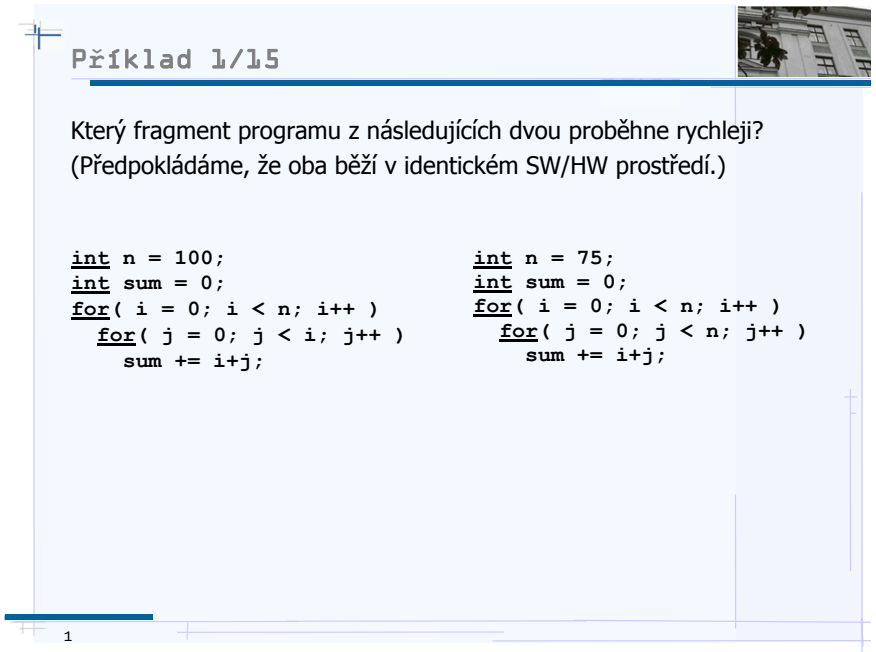
Příklad 1/15

Který fragment programu z následujících dvou proběhne rychleji?
(Předpokládáme, že oba běží v identickém SW/HW prostředí.)

```
int n = 100;
int sum = 0;
for( i = 0; i < n; i++ )
  for( j = 0; j < i; j++ )
    sum += i+j;
```

```
int n = 75;
int sum = 0;
for( i = 0; i < n; i++ )
  for( j = 0; j < n; j++ )
    sum += i+j;
```

PŘÍKLAD 1



$$\sum_{i=0}^{99} i = \frac{99(0+100)}{2} = 4950 \leq \sum_{i=0}^{74} 75 = 75 \cdot 75 = 5625 \qquad (1)$$

Simulace:

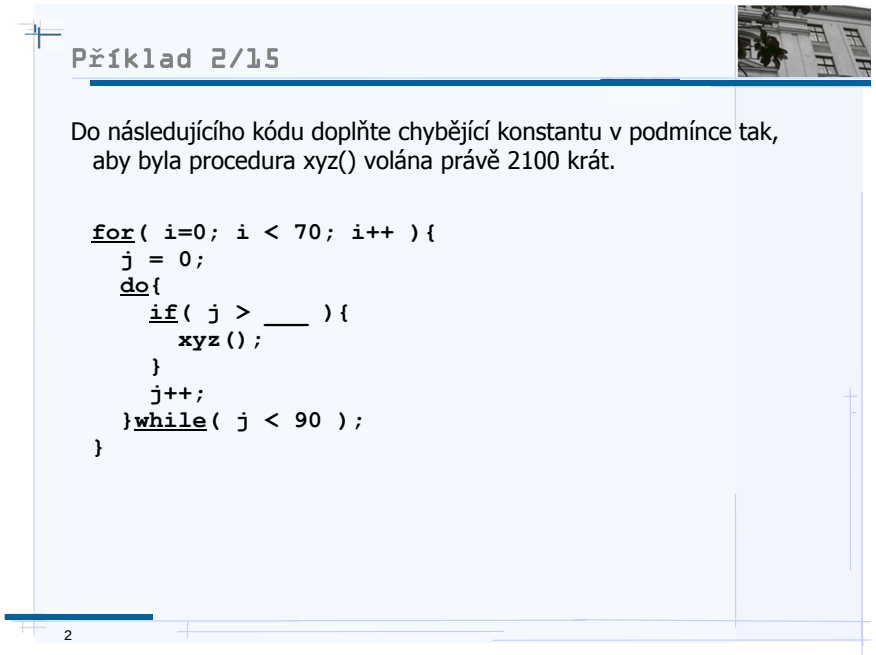
`https://onlinegdb.com/B10Gs6y2W`

Příklad 2/15

Do následujícího kódu doplňte chybějící konstantu v podmínce tak, aby byla procedura xyz() volána právě 2100 krát.

```
for( i=0; i < 70; i++ ){
  j = 0;
  do{
    if( j > ___ ){
      xyz();
    }
    j++;
  }while( j < 90 );
}
```

PŘÍKLAD 2



Příklad 2/15

Do následujícího kódu doplňte chybějící konstantu v podmínce tak, aby byla procedura xyz() volána právě 2100 krát.

```
for( i=0; i < 70; i++ ){
  j = 0;
  do{
    if( j > ___ ){
      xyz();
    }
    j++;
  }while( j < 90 );
}
```

2

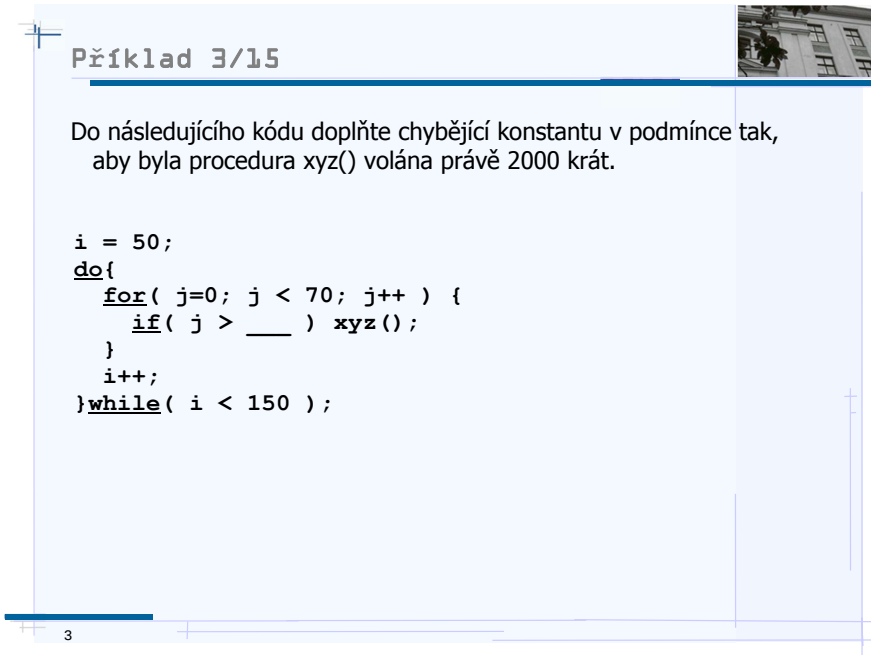
Simulace:
https://onlinegdb.com/Bysg36kh-

Příklad 3/15

Do následujícího kódu doplňte chybějící konstantu v podmínce tak, aby byla procedura xyz() volána právě 2000 krát.

```
i = 50;
do{
  for( j=0; j < 70; j++ ) {
    if( j > ___ ) xyz();
  }
  i++;
}while( i < 150 );
```

PŘÍKLAD 3



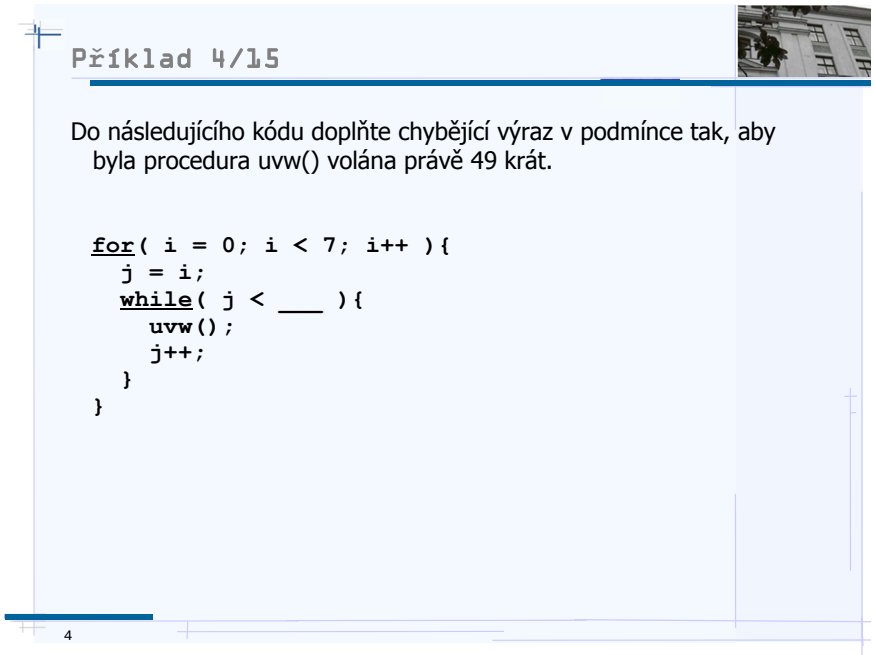
Simulace:
`https://onlinegdb.com/B1WXapy2W`

Příklad 4/15

Do následujícího kódu doplňte chybějící výraz v podmínce tak, aby byla procedura uvw() volána právě 49 krát.

```
for( i = 0; i < 7; i++ ){
  j = i;
  while( j < ___ ){
    uvw();
    j++;
  }
}
```

PŘÍKLAD 4



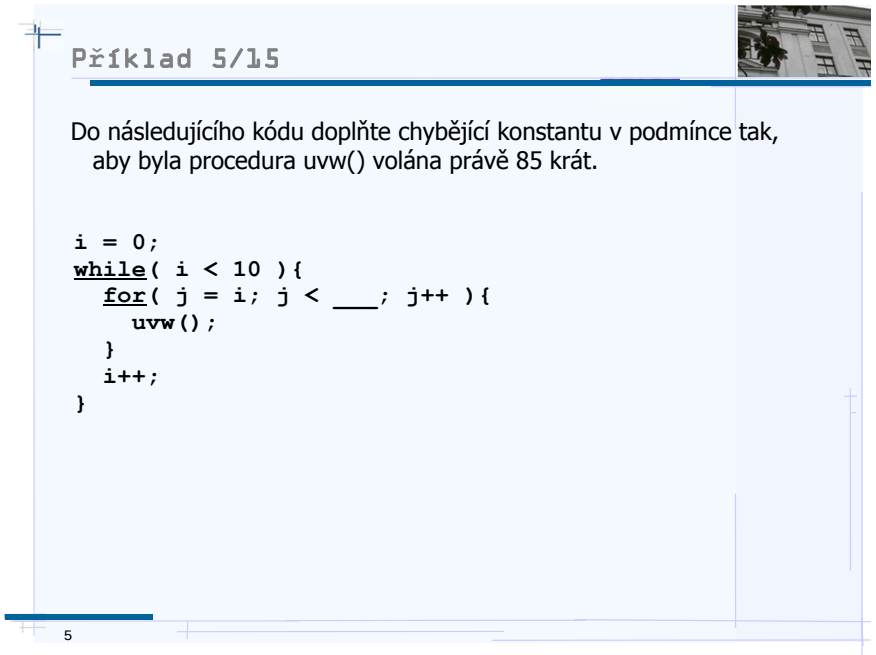
Simulace:
`https://onlinegdb.com/ByEGxRx3W`

Příklad 5/15

Do následujícího kódu doplňte chybějící konstantu v podmínce tak, aby byla procedura uvw() volána právě 85 krát.

```
i = 0;
while( i < 10 ){
  for( j = i; j < ___; j++ ){
    uvw();
  }
  i++;
}
```

PŘÍKLAD 5



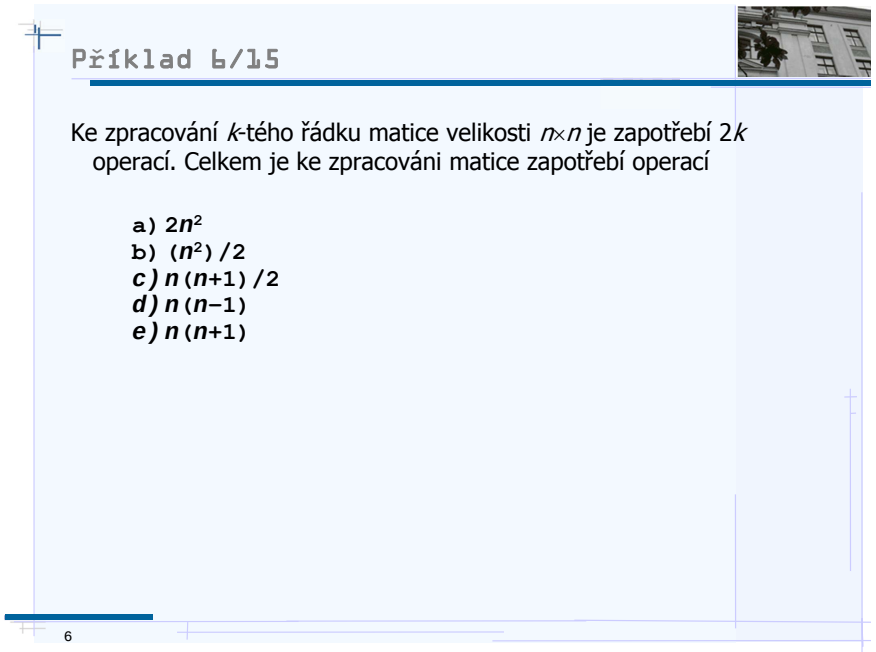
Simulace:
`https://onlinegdb.com/By56--Z3Z`

Příklad 6/15

Ke zpracování k-tého řádku matice velikosti $n \times n$ je zapotřebí $2k$ operací. Celkem je ke zpracováni matice zapotřebí operací

a) $2n^2$
b) $(n^2)/2$
c) $n(n+1)/2$
d) $n(n-1)$
e) $n(n+1)$

Příklad 6



$$2+4+6+8+\ldots+2n = 2\cdot(1+2+3+4+\ldots+n) =$$
$$= 2\cdot\frac{n(1+n)}{2} = n(n+1)$$

Příklad 7/15

A.

Úloha, jejíž doba řešení je $C \cdot n^2$, kde n je rozsah vstupních dat, se řeší na počítači pro n = 5000. Je zakoupen nový počítač, který je cca 2.5 krát rychlejší. Jak je možno zvětšit rozsah vstupních dat, aby byla úloha vyřešena na novém počítači ve stejném čase?

B.

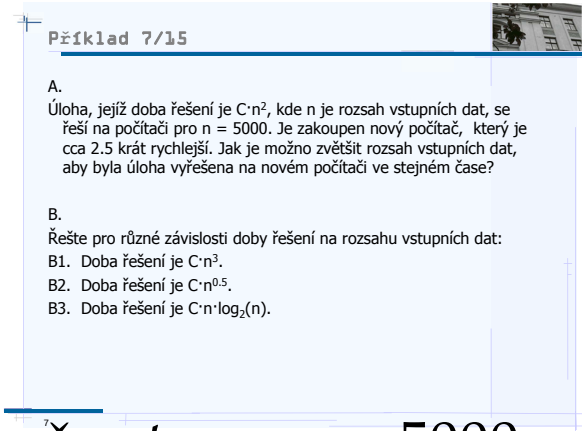
Řešte pro různé závislosti doby řešení na rozsahu vstupních dat:

B1. Doba řešení je $C \cdot n^3$.

B2. Doba řešení je $C \cdot n^{0.5}$.

B3. Doba řešení je $C \cdot n \cdot \log_2(n)$.

PŘÍKLAD 7A


Příklad 7/15

A.
Úloha, jejíž doba řešení je $C \cdot n^2$, kde n je rozsah vstupních dat, se řeší na počítači pro n = 5000. Je zakoupen nový počítač, který je cca 2.5 krát rychlejší. Jak je možno zvětšit rozsah vstupních dat, aby byla úloha vyřešena na novém počítači ve stejném čase?

B.
Řešte pro různé závislosti doby řešení na rozsahu vstupních dat:
B1. Doba řešení je $C \cdot n^3$.
B2. Doba řešení je $C \cdot n^{0.5}$.
B3. Doba řešení je $C \cdot n \cdot \log_2(n)$.

Starý počítač udělá za čas t pro $n = 5000$ operací:

$$C \cdot 5000^2$$

Nový udělá za stejný čas t operací $2.5\times$ více.

$$C \cdot 5000^2 \cdot 2.5$$

Což odpovídá zpracování vstupních dat o velikosti n_2:

$$C \cdot 5000^2 \cdot 2.5 = C \cdot n_2^2$$

$$n_2 = 5000\sqrt{2.5}$$

$$n_2 \approx 7905.69$$

Příklad 7/15

A.

Úloha, jejíž doba řešení je $C \cdot n^2$, kde n je rozsah vstupních dat, se řeší na počítači pro n = 5000. Je zakoupen nový počítač, který je cca 2.5 krát rychlejší. Jak je možno zvětšit rozsah vstupních dat, aby byla úloha vyřešena na novém počítači ve stejném čase?

B.

Řešte pro různé závislosti doby řešení na rozsahu vstupních dat:

B1. Doba řešení je $C \cdot n^3$.

B2. Doba řešení je $C \cdot n^{0.5}$.

B3. Doba řešení je $C \cdot n \cdot \log_2(n)$.

Příklad 7B

$$C \cdot 5000^3 \cdot 2.5 = C \cdot n_{\mathrm{B1}}^3$$
$$n_{\mathrm{B1}} = 5000\sqrt[3]{2.5}$$
$$n_{\mathrm{B1}} \approx 6786.04$$

$$C \cdot 5000^{0.5} \cdot 2.5 = C \cdot n_{\mathrm{B2}}^{0.5}$$
$$n_{\mathrm{B2}} = 5000 \cdot 2.5^2$$
$$n_{\mathrm{B2}} \approx 31250$$

$$C \cdot 5000 \log_2(5000) \cdot 2.5 = C \cdot n_{\mathrm{B3}} \log_2(n_{\mathrm{B3}})$$
$$n_{\mathrm{B3}} \approx 11397.4$$

https://www.wolframalpha.com/input/?i=5000+log2(5000)+*+2.5+%3D+n+log2(n)

Příklad 8/15

A.

Stroj provádí 10^9 operací za sekundu. Pro výpočet je k dispozici 1 hodina. Určete, jaká může být maximální hodnota n, která určuje velikost vstupních dat, v případě že počet nezbytných instrukcí pro zpracování dat o velikosti n je $n^{3/2}$.

B.

Určete maximální hodnotu n pro počet nezbytných instrukcí:

B1 $n^{5/4}$.

B2 $n \cdot \log_2(n) \cdot \log_2(\log_2(n))$.

B3. $n^2 \cdot \log_2(n)$.

Příklad 8A

Operací za hodinu

$$10^9 \cdot 3600 = 3.6 \cdot 10^{12} \tag{2}$$

Možná velikost vstupu:

$$3.6 \cdot 10^{12} = n^{\frac{3}{2}}$$

$$n = \left(3.6 \cdot 10^{12}\right)^{\frac{2}{3}}$$

$$n \approx 2.3489 \cdot 10^8$$

Příklad 8/15

A.

Stroj provádí 10^9 operací za sekundu. Pro výpočet je k dispozici 1 hodina. Určete, jaká může být maximální hodnota n, která určuje velikost vstupních dat, v případě že počet nezbytných instrukcí pro zpracování dat o velikosti n je $n^{3/2}$.

B.

Určete maximální hodnotu n pro počet nezbytných instrukcí:

B1 $n^{5/4}$.

B2 $n \cdot \log_2(n) \cdot \log_2(\log_2(n))$.

B3. $n^2 \cdot \log_2(n)$.

Příklad 8B

$$3.6 \cdot 10^{12} = n_{\mathrm{B1}}^{\frac{5}{4}}$$

$$n_{\mathrm{B1}} = \left(3.6 \cdot 10^{12}\right)^{\frac{4}{5}}$$

$$n_{\mathrm{B1}} \approx 1.10928 \cdot 10^{10}$$

$$3.6 \cdot 10^{12} = n_{\mathrm{B2}} \cdot \log_2(n_{\mathrm{B2}}) \cdot \log_2\left(\log_2(n_{\mathrm{B2}})\right)$$

$$n_{\mathrm{B2}} \approx 2.06080 \cdot 10^{10}$$

https://www.wolframalpha.com/input/?i=10%5E9*3600+%3D+n+log2(n)+*+log2(log2(n))

$$3.6 \cdot 10^{12} = n_{\mathrm{B3}}^2 \cdot \log_2(n_{\mathrm{B2}})$$

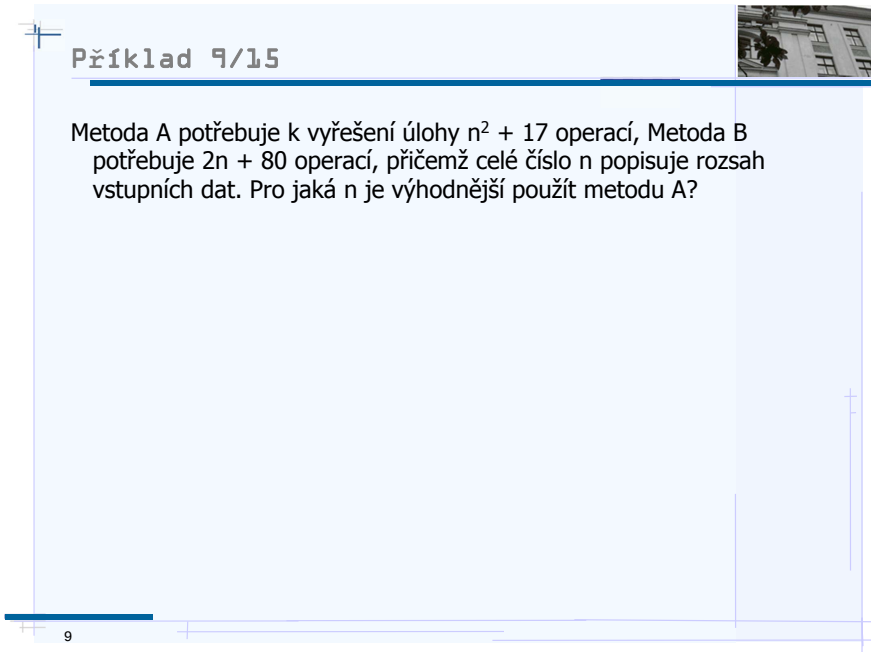
$$n_{\mathrm{B3}} \approx 4.38277 \cdot 10^5$$

https://www.wolframalpha.com/input/?i=10%5E9*3600+%3D+n%5E2+log2(n)

Příklad 9/15

Metoda A potřebuje k vyřešení úlohy $n^2 + 17$ operací, Metoda B potřebuje $2n + 80$ operací, přičemž celé číslo n popisuje rozsah vstupních dat. Pro jaká n je výhodnější použít metodu A?

Příklad 9



$$n^2 + 17 < 2n + 80$$

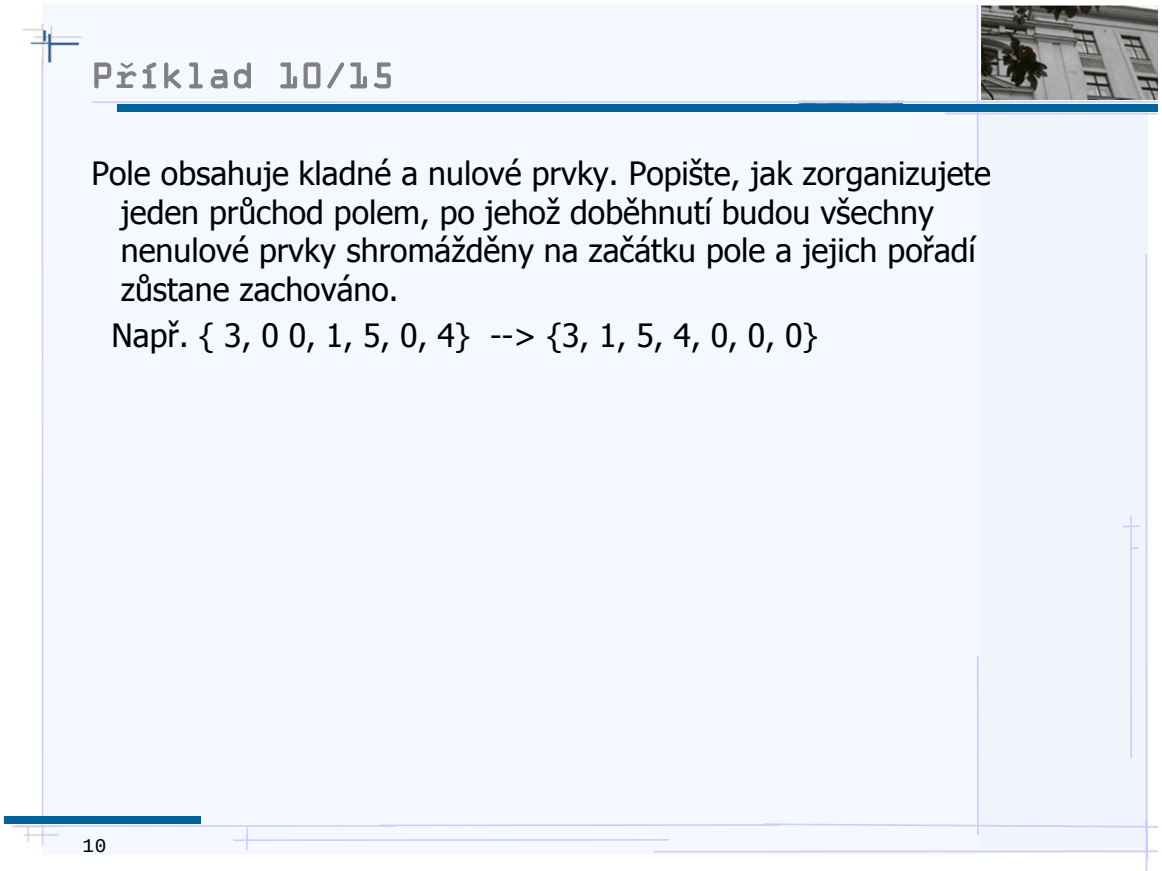
https://www.wolframalpha.com/input/?i=n%5E2%2B17+%3C+2n+%2B+80

Příklad 10/15

Pole obsahuje kladné a nulové prvky. Popište, jak zorganizujete jeden průchod polem, po jehož doběhnutí budou všechny nenulové prvky shromážděny na začátku pole a jejich pořadí zůstane zachováno.

Např. { 3, 0 0, 1, 5, 0, 4} --> {3, 1, 5, 4, 0, 0, 0}

Příklad 10

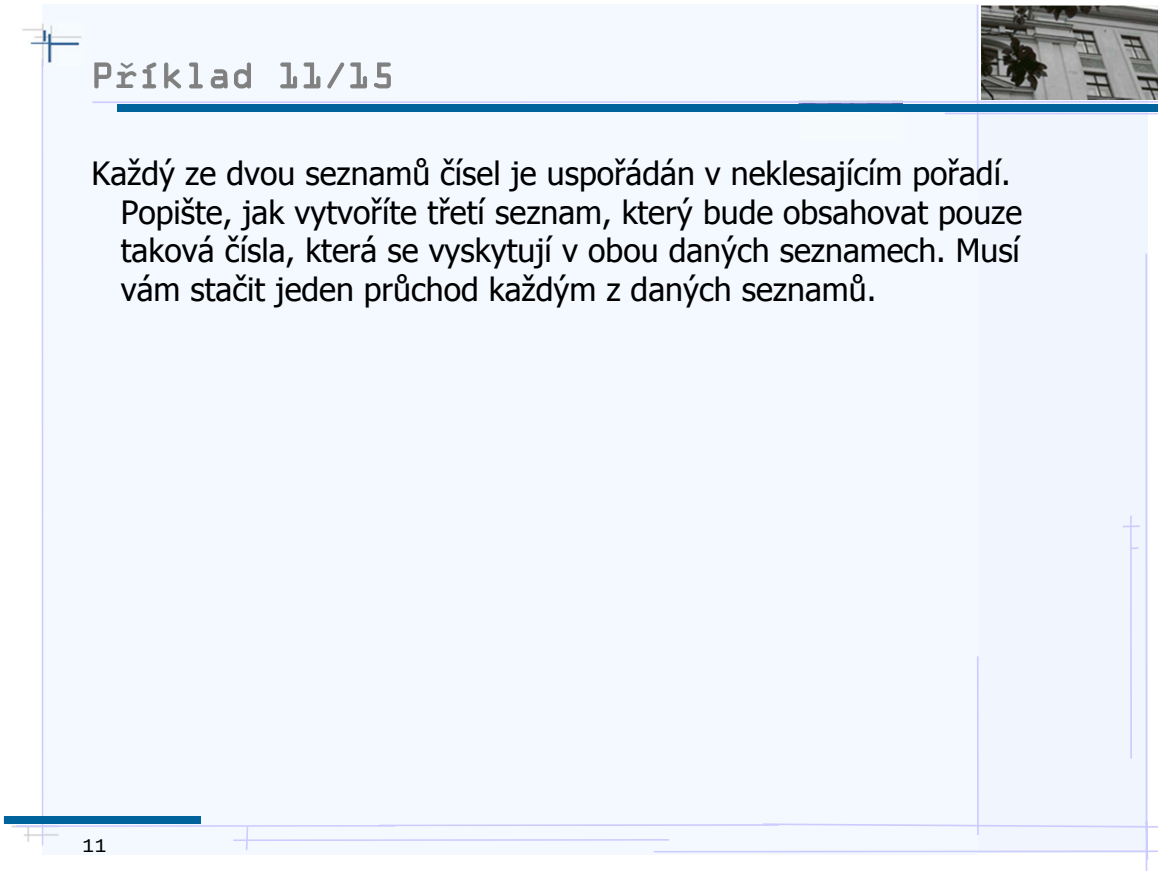


https://trinket.io/python/aab6486ab2

Příklad 11/15

Každý ze dvou seznamů čísel je uspořádán v neklesajícím pořadí. Popište, jak vytvoříte třetí seznam, který bude obsahovat pouze taková čísla, která se vyskytují v obou daných seznamech. Musí vám stačit jeden průchod každým z daných seznamů.

PŘÍKLAD 11

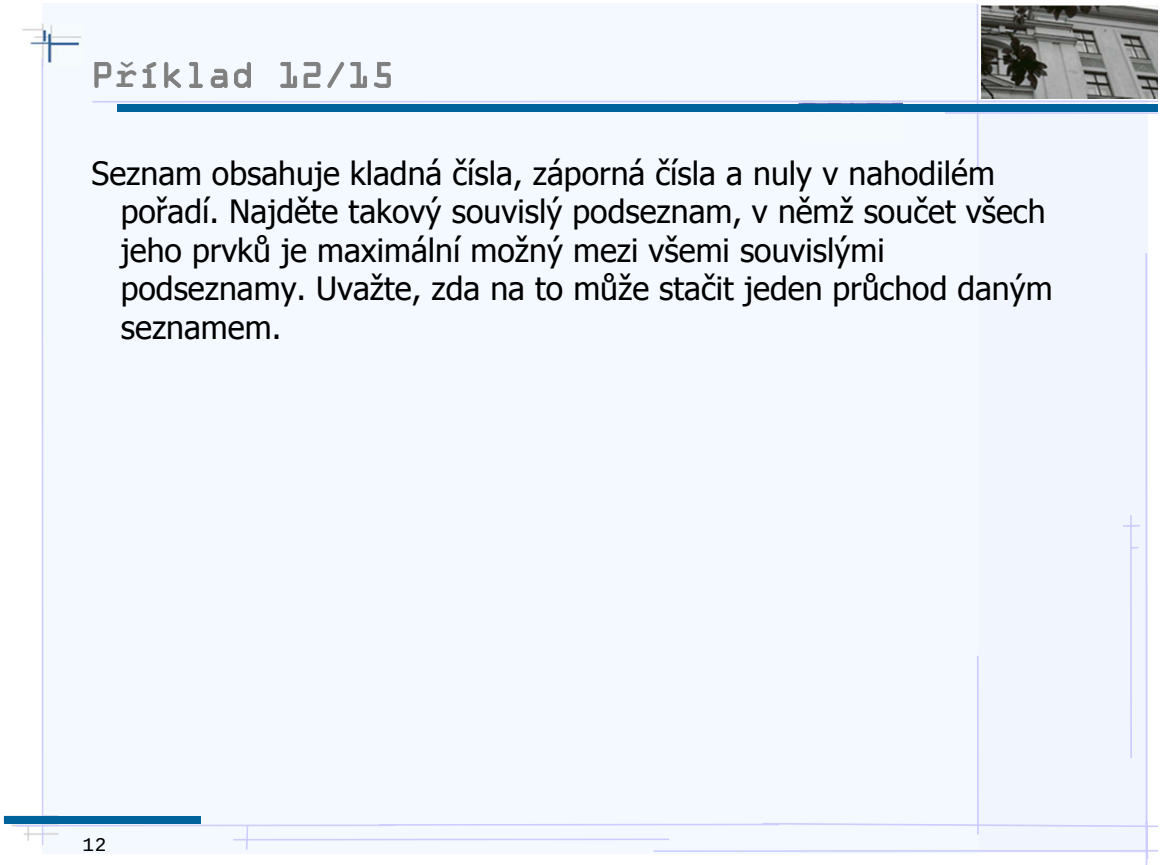


https://trinket.io/python/5040d42d6c

Příklad 12/15

Seznam obsahuje kladná čísla, záporná čísla a nuly v nahodilém pořadí. Najděte takový souvislý podseznam, v němž součet všech jeho prvků je maximální možný mezi všemi souvislými podseznamy. Uvažte, zda na to může stačit jeden průchod daným seznamem.

Příklad 12



https://en.wikipedia.org/wiki/Maximum_subarray_problem http://www.geeksforgeeks.org/largest-sum-contiguous-subarray/

Příklad 13/15

Seznam obsahuje N+1 celých čísel, každé leží v intervalu <1,N>, čísla nejsou v seznamu nijak uspořádána. Víme, že v seznamu se vyskytuje jedno číslo dvakrát, ostatní jen jednou. Určete duplikované číslo. Musí vám stačit jeden průchod seznamem a konstantně velká přidaná paměť.

Příklad 13

Příklad 13/15

Seznam obsahuje N+1 celých čísel, každé leží v intervalu <1,N>, čísla nejsou v seznamu nijak uspořádána. Víme, že v seznamu se vyskytuje jedno číslo dvakrát, ostatní jen jednou. Určete duplikované číslo. Musí vám stačit jeden průchod seznamem a konstantně velká přidaná paměť.

13

https://trinket.io/python3/d306655e20

Příklad 14/15

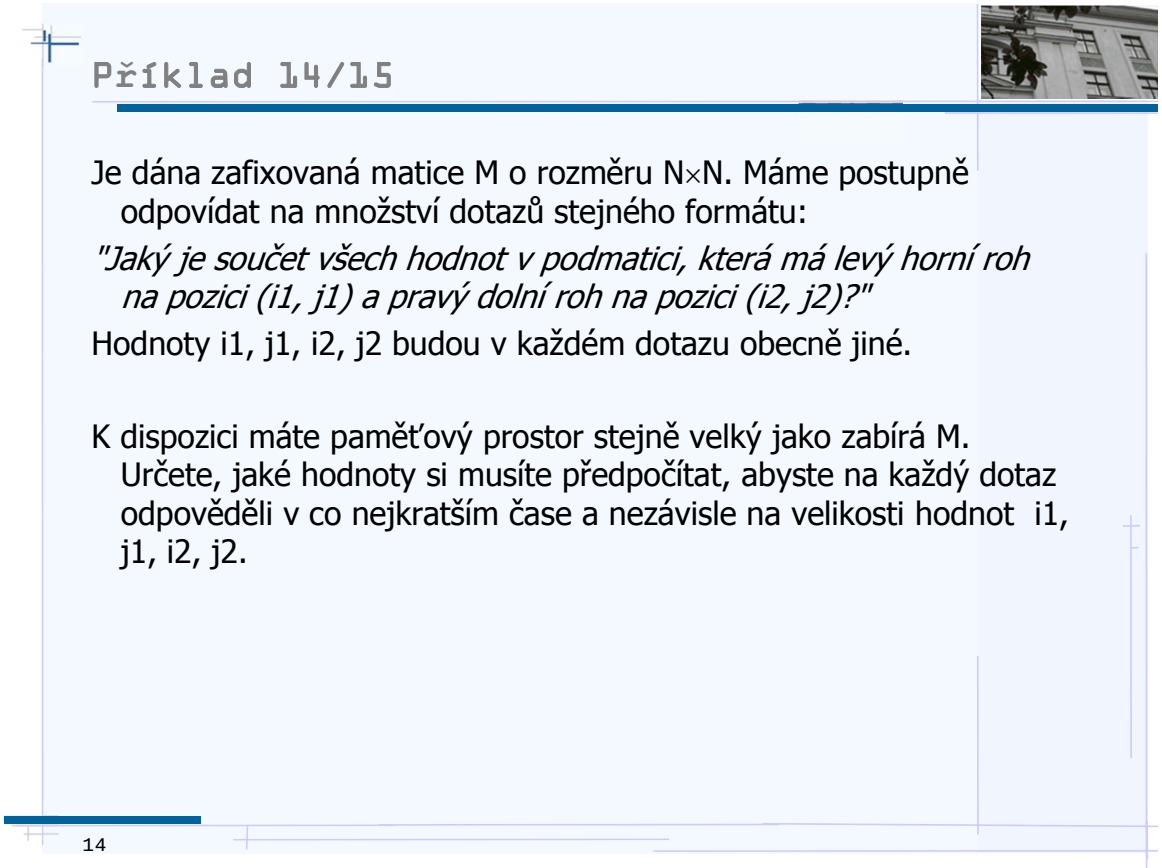
Je dána zafixovaná matice M o rozměru N×N. Máme postupně odpovídat na množství dotazů stejného formátu:

"Jaký je součet všech hodnot v podmatici, která má levý horní roh na pozici (i1, j1) a pravý dolní roh na pozici (i2, j2)?"

Hodnoty i1, j1, i2, j2 budou v každém dotazu obecně jiné.

K dispozici máte paměťový prostor stejně velký jako zabírá M. Určete, jaké hodnoty si musíte předpočítat, abyste na každý dotaz odpověděli v co nejkratším čase a nezávisle na velikosti hodnot i1, j1, i2, j2.

PŘÍKLAD 14



http://www.geeksforgeeks.org/submatrix-sum-queries/
https://i.stack.imgur.com/65t18.png