

Úvod

kuko

24.9.2025

Vybrané partie z datových štruktúr

Administrativa

- `https://kubokovac.eu/ds/`
- `jakub.kovac47@gmail.com`

Domáce úlohy a Hodnotenie

Všetky termíny sú do 8:00 ráno, t.j. pred prednáškou. Odovzdávanie mailom.

Hodnotenie: 25 bodov za D.Ú. + 15 bodov zo skúšky. **Na známku E a lepšie je potrebných 12.5 bodov z D.Ú.**

Body	Hodnotenie
40 — 34.5 points	A
34 — 30 points	B
29.5 — 23 points	C
22.5 — 17.5 points	D
17 — 12.5 points	E
< 12 points	Fx

Prednášky

Úvod, Ochtuvávka



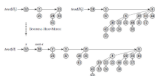
Amortizácia, vektory, stromy



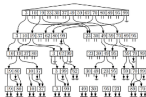
Splay stromy



Haldy



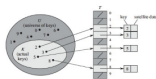
Geometria



Perzistentné dátové štruktúry



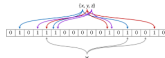
Hešovanie



Optimalizácia



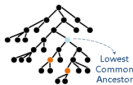
Bloomove a podielové filtre



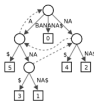
Rank/Select, Wavelet trees



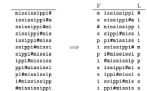
LCA a RMQ



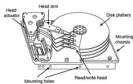
Sufixové stromy a polia



BWT a FM-index



Externé dátové štruktúry



Streamové algoritmy

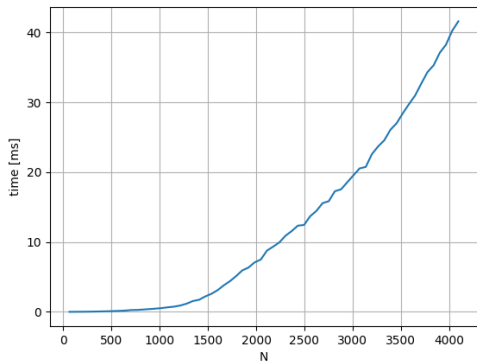


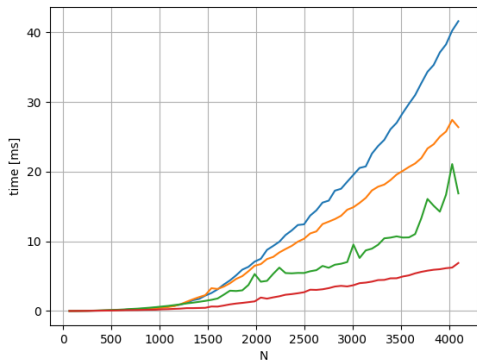
Transponovanie matice

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}^T$$

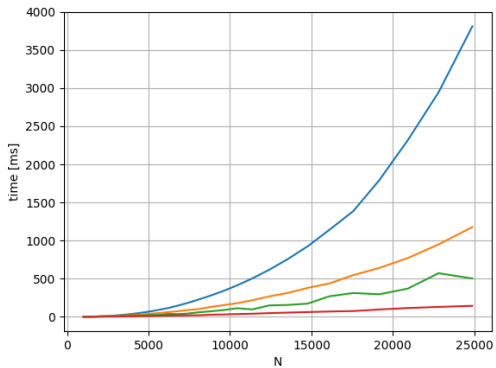
```
const int N = 1000;
int A[N][N];

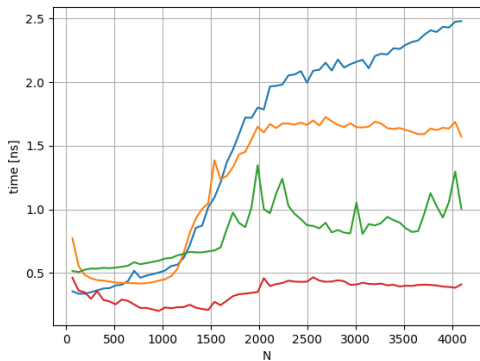
void transpose() {
    for (int i=0; i<N; ++i)
        for (int j=i+1; j<N; ++j)
            swap(A[i][j], A[j][i]);
}
```



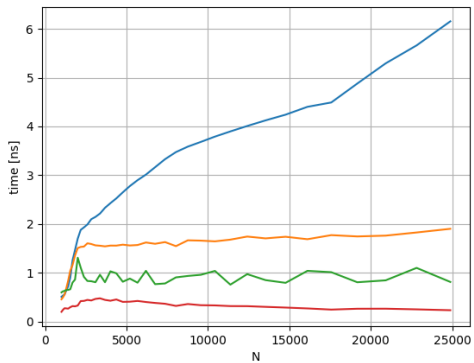


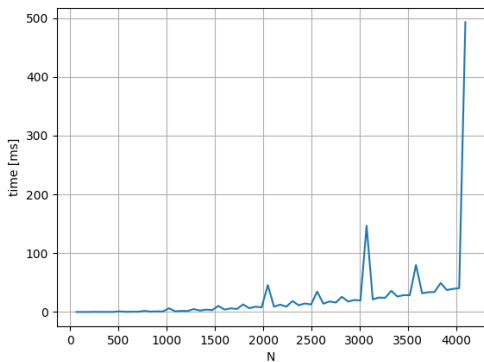
ŠOK 1: Existujú lepšie riešenia



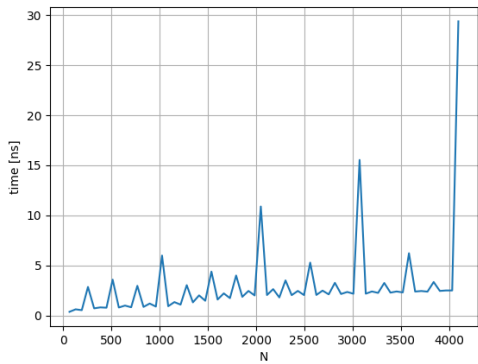


ŠOK 2: T/n^2 nie je konštanta?



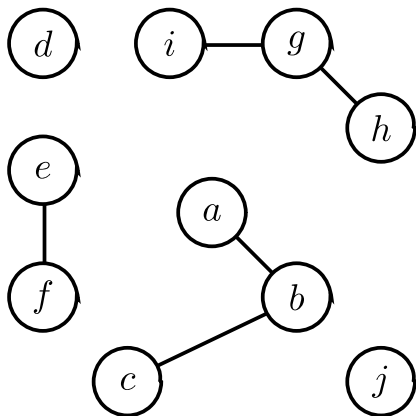


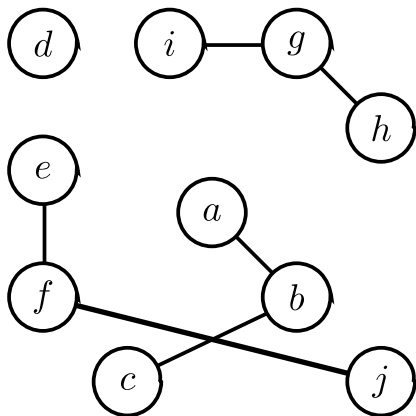
ŠOK 3: WTF?

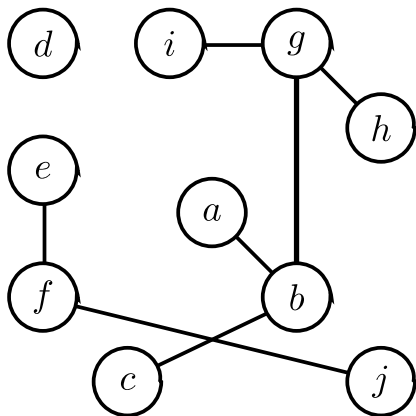


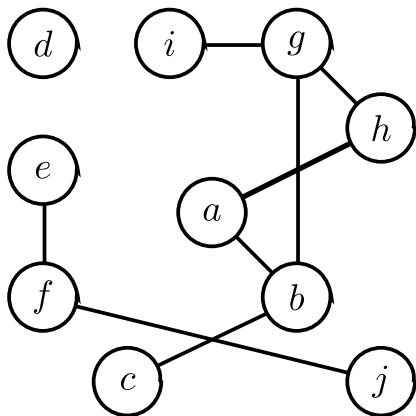
Union Find

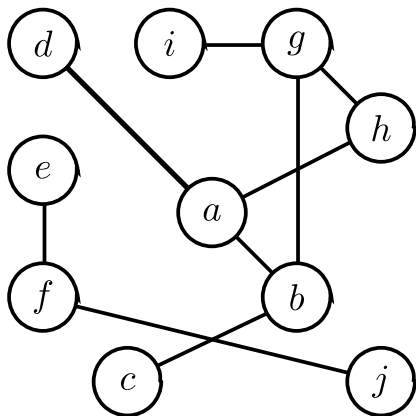
- $join(x,y)$: prepojí vrcholy x a y hranou,
- $connected(x,y)$: zjistí, či mezi vrcholmi x a y existuje cesta.











Vylepšenie #1: Union by Rank

- rank - na začiatku nula
- ak spojíme stromy s *rovnakým* rankom, koreňu zvýšime o 1
- vždy pripojíme strom s menším rankom pod strom s väčším rankom

Vylepšenie #1: Union by Rank

- strom s rankom r má výšku najviac r
- strom s rankom r má aspoň 2^r vrcholov
- teda rank r môže mať najviac $n/2^r$ vrcholov
- z toho vyplýva, že maximálny rank je $\lg n$, čo je zároveň maximálna hĺbka vrcholov a teda časová zložitosť klesne na $O(\lg n)$

Vylepšenie #1: Union by Rank

- strom s rankom r má výšku najviac r
- strom s rankom r má aspoň 2^r vrcholov
- teda rank r môže mať najviac $n/2^r$ vrcholov
- z toho vyplýva, že maximálny rank je $\lg n$, čo je zároveň maximálna hĺbka vrcholov a teda časová zložitosť klesne na $O(\lg n)$

Vylepšenie #1: Union by Rank

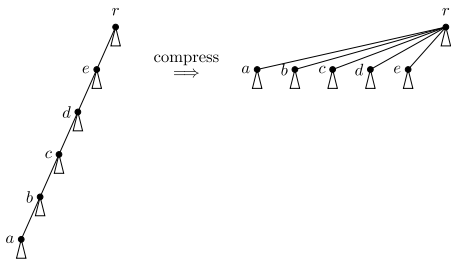
- strom s rankom r má výšku najviac r
- strom s rankom r má aspoň 2^r vrcholov
- teda rank r môže mať najviac $n/2^r$ vrcholov
- z toho vyplýva, že maximálny rank je $\lg n$, čo je zároveň maximálna hĺbka vrcholov a teda časová zložitosť klesne na $O(\lg n)$

Vylepšenie #1: Union by Rank

- strom s rankom r má výšku najviac r
- strom s rankom r má aspoň 2^r vrcholov
- teda rank r môže mať najviac $n/2^r$ vrcholov
- z toho vyplýva, že maximálny rank je $\lg n$, čo je zároveň maximálna hĺbka vrcholov a teda časová zložitosť klesne na $O(\lg n)$

Vylepšenie #2: Path Compression

Ked' $\text{find}(x)$ nájde koreň, prejdeme cestu 2x a všetky vrcholy napojíme priamo pod tento koreň.



```
int p[MAX]; // ak p[x] > 0, p[x] je otec x
            // ak p[x] <= 0, tak -p[x] je rank(x)

int find(int x) {
    if (p[x] <= 0) {
        return x; // koren
    } else {
        int root = find(p[x]); // rekurzivne
        p[x] = root;
    }
}

void union(int x, int y) { link(find(x), find(y)); }

void link(int rx, int ry) { // predpokladame, ze rx, ry su korene
    if (rx == ry) return; // a)
    if (p[rx] == p[ry]) { p[rx]=ry; p[ry]--; } // b)
    else if (-p[rx] < -p[ry]) p[rx]=ry; // c)
    else p[ry] = rx; // d)
}
```

Veta

Ak máme najviac $n < 2^{65535}$ prvkov, potom ľubovoľná postupnosť m volaní `find` vykoná najviac $6m + 2n$ skokov.

Vo všeobecnosti platí, že ľubovoľná postupnosť m operácií `union` a `find` má celkovú časovú zložitosť $O((m + n) \log^ n)$, kde \log^* je tzv. iterovaný logaritmus.*

Veta

Ak máme najviac $n < 2^{65535}$ prvkov, potom ľubovoľná postupnosť m volaní `find` vykoná najviac $6m + 2n$ skokov.

Vo všeobecnosti platí, že ľubovoľná postupnosť m operácií `union` a `find` má celkovú časovú zložitosť $O((m + n) \log^ n)$, kde \log^* je tzv. iterovaný logaritmus.*

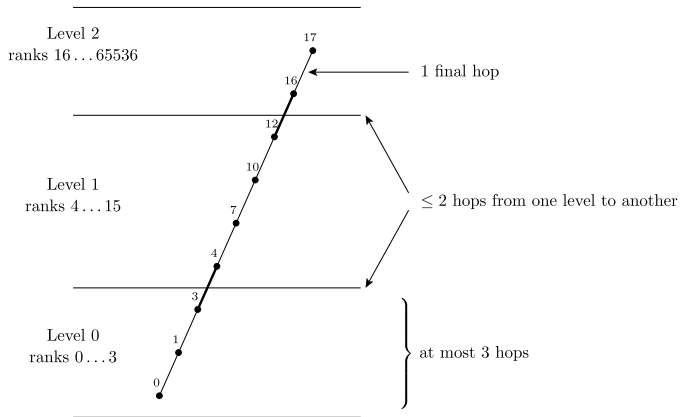
- vysoký rank je zriedkavý: iba $n/2^r$ vrcholov dosiahne rank r
- ranky na ceste ku koreňom stúpajú – $rank(p(x)) > rank(x)$
- rank vrcholu môže stúpať len kým je koreň; po napojení pod iný vrchol sa už rank nemení
- rank rodiča môže ďalej rásť (napr. pri kompresii cesty vrchol napojíme na vyššieho predka)
- def. $\Delta(x) = rank(p(x)) - rank(x)$

- vysoký rank je zriedkavý: iba $n/2^r$ vrcholov dosiahne rank r
- ranky na ceste ku koreňom stúpajú – $rank(p(x)) > rank(x)$
- rank vrcholu môže stúpať len kým je koreň; po napojení pod iný vrchol sa už rank nemení
- rank rodiča môže ďalej rásť (napr. pri kompresii cesty vrchol napojíme na vyššieho predka)
- def. $\Delta(x) = rank(p(x)) - rank(x)$

- vysoký rank je zriedkavý: iba $n/2^r$ vrcholov dosiahne rank r
- ranky na ceste ku koreňom stúpajú – $rank(p(x)) > rank(x)$
- rank vrcholu môže stúpať len kým je koreň; po napojení pod iný vrchol sa už rank nemení
- rank rodiča môže ďalej rásť (napr. pri kompresii cesty vrchol napojíme na vyššieho predka)
- def. $\Delta(x) = rank(p(x)) - rank(x)$

- vysoký rank je zriedkavý: iba $n/2^r$ vrcholov dosiahne rank r
- ranky na ceste ku koreňom stúpajú – $rank(p(x)) > rank(x)$
- rank vrcholu môže stúpať len kým je koreň; po napojení pod iný vrchol sa už rank nemení
- rank rodiča môže ďalej rásť (napr. pri kompresii cesty vrchol napojíme na vyššieho predka)
- def. $\Delta(x) = rank(p(x)) - rank(x)$

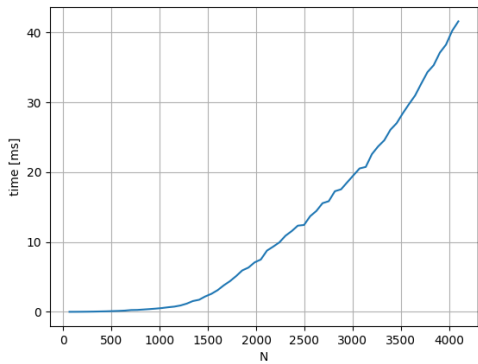
- vysoký rank je zriedkavý: iba $n/2^r$ vrcholov dosiahne rank r
- ranky na ceste ku koreňom stúpajú – $rank(p(x)) > rank(x)$
- rank vrcholu môže stúpať len kým je koreň; po napojení pod iný vrchol sa už rank nemení
- rank rodiča môže ďalej rásť (napr. pri kompresii cesty vrchol napojíme na vyššieho predka)
- def. $\Delta(x) = rank(p(x)) - rank(x)$

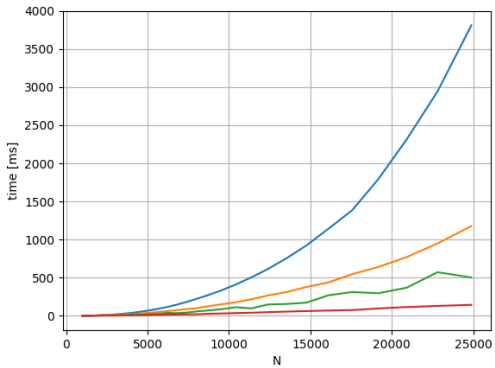


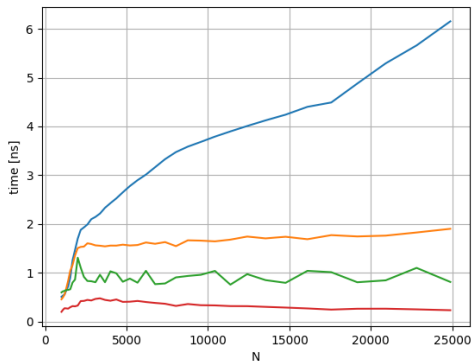
Transponovanie matice

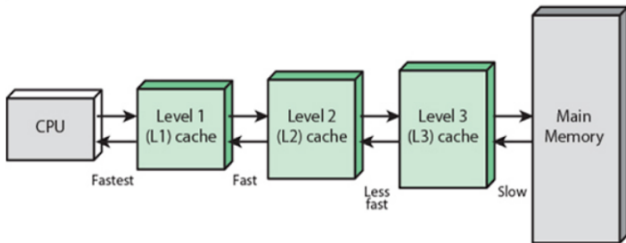
```
const int N = 1000;
int A[N][N];

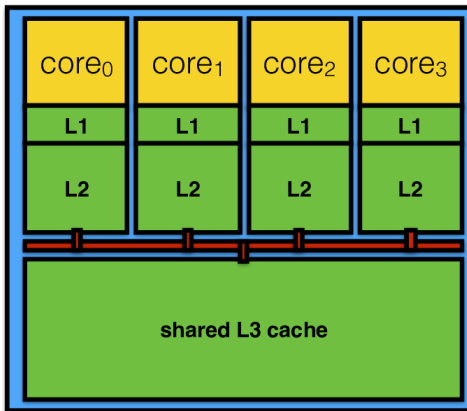
void transpose() {
    for (int i=0; i<N; ++i)
        for (int j=i+1; j<N; ++j)
            swap(A[i][j], A[j][i]);
}
```

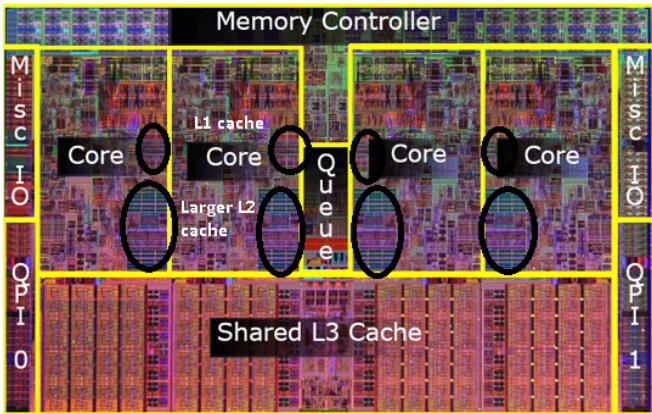




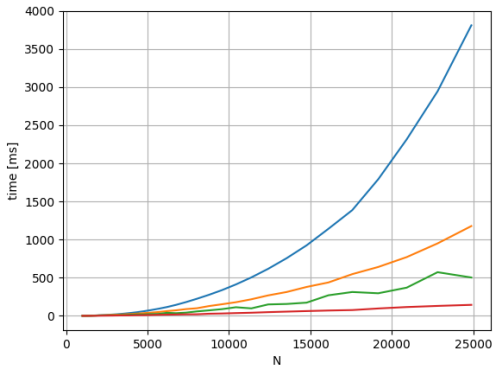








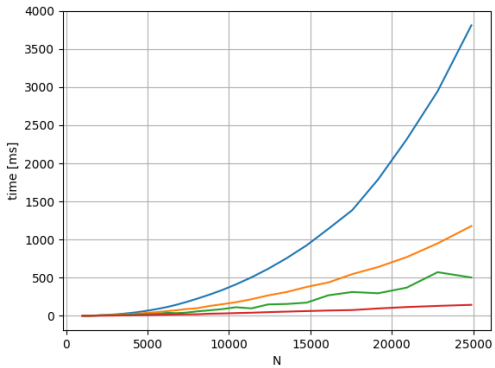
```
void transpose_block(vector<vector<int>> &A) {
    int N = A.size();
    int B = 64;
    for (int k = 0; k < N; k += B) {
        for (int i = k; i < k + B && i < N; ++i)
            for (int j = k + 1; j < k + B && j < N; ++j)
                swap(A[i][j], A[j][i]);
        for (int l = k + B; l < N; l += B)
            for (int i = k; i < k + B && i < N; ++i)
                for (int j = l; j < l + B && j < N; ++j)
                    swap(A[i][j], A[j][i]);
    }
}
```



```

void transpose_block2(vector<vector<int>> &A) {
    int N = A.size();
    int B = 1040;
    int b = 4;
    for (int x = 0; x < N; x += B) {
        for (int k = x; k < x + B && k < N; k += b) {
            for (int i = k; i < k + b && i < N; ++i)
                for (int j = k + 1; j < k + b && j < N; ++j)
                    swap(A[i][j], A[j][i]);
            for (int l = k + b; l < x + B && l < N; l += b)
                for (int i = k; i < k + b && i < N; ++i)
                    for (int j = l; j < l + b && j < N; ++j)
                        swap(A[i][j], A[j][i]);
        }
    }
    for (int y = x + B; y < N; y += B)
        for (int k = x; k < x + B && k < N; k += b)
            for (int l = y; l < y + B && l < N; l += b)
                for (int i = k; i < k + b && i < N; ++i)
                    for (int j = l; j < l + b && j < N; ++j)
                        swap(A[i][j], A[j][i]);
}
}

```



```

void transpose_rec(const int N, vector<vector<int>> &A,
                  int B, int i0, int j0, int i1, int j1) {
    if (B <= 4) {
        if (i0 == j0) {
            for (int i = i0; i < i0 + B && i < N; ++i)
                for (int j = i + 1; j < i0 + B && j < N; ++j)
                    swap(A[i][j], A[j][i]);
        } else {
            for (int i = i0, jj = j1; i < i0 + B && i < N && jj < N; ++i, ++jj)
                for (int j = j0, ii = i1; j < j0 + B && j < N && ii < N; ++j, ++ii)
                    swap(A[i][j], A[ii][jj]);
        }
    } else {
        int h = B / 2;
        transpose_rec(N, A, h, i0, j0, i1, j1);
        transpose_rec(N, A, B - h, i0 + h, j0 + h, i1 + h, j1 + h);
        transpose_rec(N, A, B - h, i0 + h, j0, i1, j1 + h);
    }
}

transpose_rec(N, A, N, 0, 0, 0, 0);

```

