Zadání Máme dána čísla $m, n \in \mathbb{N}$ a matici $A \in \{0,1\}^{m \times n}$. Navrhněte algoritmus, který najde největší podmatici obsahující pouze 1

Popis řešení Nejprve vytvoříme pomocné matice $D, U \in \mathbb{N}^{m \times n}$, kde

$$U_{i,j} = A_{i,j} \cdot (i - i' + 1), \text{ kde } i' = \min \{m+1, \ell \mid \forall k \in \{\ell, \ldots, i\}\ A_{k,j} = 1\}.$$

Jinými slovy, $U_{i,j}$ je délka souvislého úseku jedniček nad (i,j), pokud $A_{i,j}$ je 1. Jinak $U_{i,j} = 0$. Podobně definujeme matici D jako délku souvislého úseku 1 směrem dolů od (i,j). Tedy

$$D_{i,j} = A_{i,j} \cdot (i' - i + 1), \text{ kde } i' = \max \{0, \ell \mid \forall k \in \{i, \ldots, \ell\}\ A_{k,j} = 1\}.$$

Následně učiníme pozorování, že každá největší podmatice jedniček sousedí v nějakém řádku nalevo s nulou nebo s okrajem matice (speciálně pokud naši matici obalíme ze všech stran nulami, víme, že sousedí s nulou). Kdyby nesousedila se žádnou nulou můžeme ji rozšířit o jeden sloupec doleva. V našem algoritmu tedy budeme hledat největší souvislou podmatici, pomocí hledání nuly, která s ní sousedí nalevo.

Následující algoritmus předpokládá, že na vstupu dostane matici A, jejíž nultý sloupec, nultý řádek, $n+1$ sloupec a $m+1$ řádek je složen ze samých nul.

Pseudokód

Algorithm 1: Jedničková podmatice

```
1  matrix precompute_U(m, n, A):
2      U = 0^{m×n};
3      for i ∈ {1, ..., m} do
4          for j ∈ {1, ..., n} do
5              U_{i,j} = A_{i,j} · (U_{i−1,j} + 1);
6          end
7      end
8      return U;
9  matrix precompute_D(m, n, A):
10     D = 0^{m×n};
11     for i ∈ {m, ..., 1} do
12         for j ∈ {n, ..., 1} do
13             D_{i,j} = A_{i,j} · (D_{i+1,j} + 1);
14         end
15     end
16     return D;
17 int main(m, n, A):
18     U =precompute_U(m, n, A);
19     D =precompute_D(m, n, A);
20     max_up = ∞;
21     max_down = ∞;
22     max_size = 0;
23     c = 0;
24     for i ∈ {1, ... m} do
25         for j ∈ {1, ... n} do
26             if A_{i,j} = 0 then
27                 max_up = ∞;
28                 max_down = ∞;
29                 c = 0;
30             end
31             else
32                 max_up = min(max_up, U_{i,j});
33                 max_down = min(max_down, D_{i,j});
34                 c = c + 1;
                   /* Odečítáme 1 kvůli tomu, že aktuální řádek
                      je započítán i v U i v D                    */
35                 max_size = max(max_size, c · (max_up + max_down − 1));
36             end
37         end
38     end
39     return max_size;
```

Důkaz správnosti Vychází z pozorování, že pokud je podmatice největší, sousedí na každé straně s nulou (popř. okrajem matice). Algoritmus postupně hledá jedničku v podmatici, od níž nalevo je 0 nebo okraj matice. Od takové jedničky rozšiřuje matici směrem doprava a přepočítává, jak moc můžeme matici rozšířit směrem nahoru a jak moc ji můžeme rozšířit směrem dolů. Z těchto matic bere maxima.

Prostorová složitost Potřebujeme si pamatovat 3 matice a konstantně mnoho čísel tedy $\mathcal{O}(mn)$.

Časová složitost Všechny funkce precompute_U, precompute_D a main počítají v $\mathcal{O}(mn)$. Tedy složitost je lineární ve velikosti matice $\mathcal{O}(mn)$.