

rakningarvensl

summur af eftirfarandi hafa sömu útkomu,  $\left(\frac{n(n+1)}{2}\right)$  sem er  $O(n^2)$

$$\sum_{k=0}^n k = 0 + 1 + 2 + \dots + n = O(n^2)$$
$$\sum_{k=0}^n n - k = n + (n - 1) + (n - 2) + \dots + 0 = O(n^2)$$

summur kvótaraða hafa líka sína eigin formúlu,

$$\sum_{k=0}^n r^k = \begin{cases} r \neq 1 := \frac{r^{n+1}-1}{r} - 1 \\ r = 1 := (n + 1) \end{cases}$$

logaritmar

$$x^{\log_x(y)} = y \quad x^{\log_x(y)+1} = x^{\log_x(y)} \cdot x^1 = x^{\log_x(y)} \cdot x = x^{\log_x(x \times y)} = x \times y$$

ef hægt er að mynda  $x$  sem  $z$  í veldi  $y$  þá gengur eftirfarandi  
 $x^{\log_z(n)} = (z^y)^{\log_z(n)} = z^{y \times \log_z(n)} = z^{\log_z(n^y)} = n^y$

master theorem (recursion)

$$T(n) = a \times T\left(\frac{n}{b}\right) + f(n)$$

- 1. Ef  $f(n)$  er  $O(n^c)$ , þar sem  $c$  er ehv fasti, og  $f(n)$  er minna en  $n^{\log_b(a)}$  þá er  $T(n) = O(n^{\log_b(a)})$
- 2. Ef  $f(n)$  er  $O(n^c)$  og  $f(n) = n^{\log_b(a)}$  þá er  $T(n) = O(n^{\log_b(a)} \times \log(n))$
- 3. Ef  $f(n)$  er  $O(n^c)$  og  $f(n)$  er stærra en  $n^{\log_b(a)}$  og það er til ehv fasti  $d$  sem sýnir að  $a \times f\left(\frac{n}{b}\right) \leq d \times f(n)$  fyrir öll nógu stór  $n$ , þá er tími reikniritisins  $T(n) = O(f(n))$

DYP (dynamic programming)

setjum fram dæmi:

við höfum runu spilapeninga  $t_1, ..., t_n$ . hver spilapeningur hefur tvo eiginleika, virði og lit. virði er tala á forminu 2, 4, 8, ..., 2048 og litir eru R, G eða B. Við getum myndað vensl á milli peninga ef þeir uppfylla eftirfarandi reglur:

- þeir hafa virði, eða
- þeir eru í sama lit, eða
- annar þeirra hefur virði 2

við viljum finna lengstu hlutrunu spilapeninga  $t_{i_1}...t_{i_k}$  þar sem  $i_1 < ... < i_k$ .

Tökum sem dæmi rununa  $4R, 4B, 16R, 16G, 2R$

skref

- 1. lýsum undirverkefnum
  - undirverkefni hér væri að finna lengstu venslu hlutrunu frá hverju staki
- 2. lýsum ákvörðun / ágiskun í hverju undirskrefi
  - hér þyrftum við að ákvarða hvaða stak væri best að halda áfram með í rununninni
  - það er frekar augljóst að besta stakið til bæta núverandi runu er það sem hefur lengstu hlutrunu nú þegar
- 3. lýsum rakningarvenslum
  - tökum skrefin á undan og bætum við grunntilvik

$$f(i) = \begin{cases} i = n : 1 \\ i \neq n : \max(arr \in t_i \sim t_j \wedge i < j) \end{cases}$$

- 4. lýsum hvernig koma má í veg fyrir endurtekt með því að geyma niðurstöðurstöður

- hérna þurfum við að vita hvað besta runa af stökum  $A[i + 1...]$
  - við geymum þær upplýsingar í einföldu fylki þar sem hvert stak táknar lengstu runu frá því staki
5. rökstyðjum tímaflækju
- í hverju skrefi erum við að velja  $max(arr)$  og sú aðferð ítrar yfir öll stök fylkisins í  $O(n)$  tíma
  - við endurtökum þessi undirskref fyrir öll stök fylkisins
  - lokatímaflækja er  $O(n * n) = O(n^2)$

leggir og leiðindi

leggir skilgreindir sem  $[u] \rightarrow [v]$  þar sem  $u$  er upphafspunktur og  $v$  er endapunktur.

fram / trjáleggir

v er nýr þegar djúpleit á **u** hefst:

$u.pre < v.pre < v.post < u.post$

- 1. **trjáleggur** ef foreldri(v) = u:  $[u] \rightarrow [v]$
- 2. **framleggur** annars  $[u] \rightarrow [v]$

bakleggur

v er virkur þegar dýptarleit á **u** hefst:

$v.pre < u.pre < u.post < v.post \quad [v] \leftarrow [u]$

krossleggur

v búinn lokið þegar dýptarleit á **u** hefst:  $v.post < u.pre$

netareiknirit

Net G=(V,E)	Vogtölur	Reiknirit	Tímaflækja
ó/stefnt, rásað	$w \in \mathbb{R}, w \geq 0$	Dijkstra	$O(\log(V)E)$
ó/stefnt, rásað	$w \in \mathbb{R}$	Bellman-Ford	$O(VE)$
stefnt, órásað	$w \in \mathbb{R}$	SSSPDAG	$O(V + E)$

```
def dijkstra(G, start):
    dist, parent = {}, {}
    for v in G.v: dist[v]
    Q = new minPQ(start)

    while Q:
        v = Q.pop()
        for u in G[v]:
            if dist[u] > dist[v] + G[v][u]:
                dist[u] = dist[v] + G[v][u]
                parent[u] = v
            Q.push(u)
```

**MaxFlow** (G = (V,E)) O(VE)

Tekur inn flæðisnet og skilar því með hámarksflæði, passa að leggir þurfa þyngd \_\_\_\_\_


**FlowToPaths** (MaxFlow) O(E)

Tekur inn flæðisnet, býið að finna maxflow og skilar vegum sem fylgja því með tilliti til hnúta \_\_\_\_\_

**FlowToMatching** (MaxFlow) O(E)

Tekur inn maxflow og varpar yfir í spyrðingu \_\_\_\_\_

## MaximumMatching ( $G(V,E)$ )

Tekur inn net með hnúta af týpu inn-út, og skilar hámarksspyrðingu, líka hægt að fá með MaxFlow 

## MatchingToCover ( $\text{Matching}$ )

Tekur inn spyrðingu úr falli eins og MM og skilar þakningu yfir netið