Gyökös felbontás - Megoldások

szerző: Nikházy László
előadő: Gyimesi Péter

2025. szeptember 11.

Feladatok

Feladatok csoportos munkára:

A. **POI 2017 – Shipping Containers**
`https://szkopul.edu.pl/problemset/problem/oNnWY6ZuzzhvG-jCmijiXkIk/site/?key=statement`

B. **XOR and Favourite Number**
`https://codeforces.com/contest/617/problem/E`

C. **Holes**
`https://codeforces.com/contest/13/problem/E`

D. **JOI 2018 – Bitaro's Birthday**
`https://oj.uz/problem/view/JOI18_bitaro`

Lekötő feladatok:

L1. **Induced Subgraph Queries**
`https://codeforces.com/contest/2129/problem/E`

L2. **Xenia and Tree**
`https://codeforces.com/contest/342/problem/E`

L3. **IOI 2011 - Dancing Elephants**
`https://oj.uz/problem/view/IOI11_elephants`

Megoldások

POI 2017 - Shipping Containers: Megoldás

Ötlet: Két naiv megoldást kombinálunk *SQRT decomposition* módszerrel.
Naiv megoldás 1: Tároljuk a pozíciókon lévő konténerek számát.

- Egy daru műveletnél a megfelelő pozíciókat növeljük.
- $\mathcal{O}(K \cdot N/d_i)$ - nagy d_i-k esetén hatékony
- A módosítás lassú, az eredmény lekérdezése gyors

Naiv megoldás 2: A változásokat tároljuk, lépésköz (d) szerint külön nyilvántartva.

- Legyen $add[pos][d]$: mennyivel változik a d-hez tartozó sorozat a pos helyen.
- Egy daru művelet (a, ℓ, d) esetén:

  $add[a][d] \mathrel{+}= 1$ (sorozat indul itt)

  $add[a + \ell \cdot d][d] \mathrel{-}= 1$ (sorozat véget ér itt)
- Ezután prefix-összeggel kiszámoljuk: $add[i][d] += add[i-d][d]$ minden i-re. Így $add[i][d]$ = hány daru pakol konténert az i pozíción d lépésközzel.
- **Komplexitás:** frissítés $O(1)$, de az összes pozíció kiszámolása $O(N \cdot d_{max})$.

Kombinált megoldás:

Ha $d_i \geq \sqrt{N}$, alkalmazzuk a 1-es megoldást.

Ha $d_i < \sqrt{N}$, alkalmazzuk a 2-es megoldást.

Végén összeadjuk a két rész eredményeit.

Időkomplexitás: $\mathcal{O}((K+N)\sqrt{N})$. **Kód:**
`https://usaco.guide/problems/poi-2017containers/solution`

XOR and Favourite Number: Megoldás

Ötlet: Prefix-XOR tömb bevezetése, majd Mo algoritmussal (SQRT decomposition) kezeljük a lekérdezéseket.

Prefix-XOR:

- `pref[0] = 0`, `pref[i] = pref[i-1] ^ a[i]`.
- Az `a[l..r]` rész XOR-ja: `pref[r] ^ pref[l-1]`.

Lekérdezés átalakítása:

- Számoljuk az olyan (i, j) párokat ($l - 1 \leq i < j \leq r$), ahol `pref[i] ^ pref[j] == k`.
- Ekvivalens: adott `pref[j]`, kell `pref[i] == pref[j] ^ k`.

Megoldás:

- `cnt[v]` = v előfordulásainak száma az `pref` tömbben aktuális intervallumban.
- Határ mozgatásakor frissítés $O(1)$: új `pref[x]` hozzáadásakor `result += cnt[pref[x] ^ k]`.
- Mo algoritmussal az összes lekérdezés $\mathcal{O}((n+m)\sqrt{n})$ időben és $\mathcal{O}(n)$ memória használatával megoldható.

Kód: `https://codeforces.com/contest/617/submission/15550846`

Holes: Megoldás

Ötlet: A labda ugrásait *SQRT decomposition*-nel gyorsítjuk, blokkokra bontva a lyukakat.
Adatszerkezet: Minden lyukhoz tároljuk:

- `last[i]`: az a lyuk, ahonnan a labda a blokk végén kilép,
- `cnt[i]`: hány ugrást tett a labda a blokk végéig.

Blokkok felépítése:

- Az N lyukat ($O(\sqrt{N})$ méretű blokkokra osztjuk).
- Visszafelé haladva minden lyukra számoljuk: ha a labda a blokk végéig elér, `last[i]=i`, `cnt[i]=1`, különben `last[i]=last[i+power[i]]`, `cnt[i]=cnt[i+power[i]]+1`.

Lekérdezés (labda dobása):

- A labda ugrásait blokkonként összegyűjtve gyorsan számoljuk a teljes ugrásszámot.
- Végül megkapjuk a kilépő lyuk számát és az ugrások számát.

Módosítás (lyuk erő frissítése):

- Ha egy lyuk erejét megváltoztatjuk, újraszámoljuk a teljes blokkot a fent leírt módon.

Komplexitás: $\mathcal{O}((N+M)\sqrt{N})$ időben, $\mathcal{O}(N)$ memóriában.
Kód: `https://codeforces.com/contest/13/submission/338078905`

JOI 2018 – Bitaro's Party: Megoldás

Ötlet: Hosszú út keresése egy irányított, körmentes gráfban. *SQRT decomposition*-t alkalmazunk a lekérdezések számától és a blokkolt barátok számától függően.

1. **Egyszerű eset ($q = 1$):**
   - Megfordítjuk a gráfot, így a célból a kiinduló városokhoz vezető leghosszabb utat keressük.
   - DP: `dp[i]` = leghosszabb út a célvárosból i-be.
   - Frissítés: `dp[v] = max(dp[v], dp[u] + 1)` minden $u \to v$ élre.
   - Mivel $v < u$, egyetlen végigiterálás elegendő (visszafelé).
2. **SQRT decomposition:**
   - Jelöljük a blokkolt barátok számát y-nak, legyen $B = \sqrt{10^5}$.
   - Ha $y \geq B$: kevés lekérdezés, minden lekérdezésre futtatható $\mathcal{O}(n)$ DP ($\mathcal{O}(Bn)$ összesen).
   - Ha $y < B$: előfeldolgozás

     Minden városhoz tároljuk a B leghosszabb utat különböző kiindulási pontokból.

     Lekérdezéskor csak a leghosszabb utak közül választunk, amelyek nem blokkoltak.

Komplexitás: $\mathcal{O}(Bm + n \log(Bn) + Bn)$.

Kód:

`https://usaco.guide/problems/joi-2018bitaros-birthday/solution`

Lekötő feladatok

Megoldások megtalálhatók itt:

L1. `https://codeforces.com/blog/entry/145152`

L2. `https://codeforces.com/blog/entry/8800`

L3. `https://dmoj.ca/problem/ioi11p5/editorial`