Problema `startrek` – descriere soluţie

Autor: Eugen Nodea - profesor Colegiul Național „Tudor Vladimirescu” Tg Jiu

Soluţie: Lucian Bicsi

Una din soluţiile problemei este folosind metoda programării dinamice. Soluţia va consta în două parcurgeri: una de la `1` la `n`, în care aflăm numărul maxim şi a doua de la `n` la `1`, în care determinăm soluţia minim lexicografică. Vom prezenta ideea doar în cadrul parcurgerii “stânga-drepta”, cu considerenţele că cealaltă parcurgere este foarte asemănătoare.

Fie `OK[i][j] = true` dacă pe poziţia i se poate termina un bloc continuu cu valori egale cu j.
Pentru a calcula această dinamică, avem cazul de bază `OK[0][0] = true` şi, pentru `1 ≤ i, j ≤ n` vom căuta `i' < i` care respectă `p ≤ i – i' + 1 ≤ q` şi `OK[i'][j - 1] = true`. Important este ca, în plus, pentru orice poziţie între `i' + 1` şi `i` să nu existe restricţii de valori diferite de `j`. Soluţia are complexitate `O(n * maxx * q)` şi ar trebui să obţină între 30 şi 70 de puncte, unde `maxx` este numărul maxim de ani.
Să optimizăm acum soluţia de la pasul anterior. Observaţia-cheie pentru aceasta este că, pentru un `i` fixat, **`OK[i][j] = true` pe un interval compact de valori**. Cu alte cuvinte, este suficient să calculăm capetele acestui interval, pe care le vom numi de acum `Min[i]`, respectiv `Max[i]`.
Pentru a calcula aceste valori, observăm că putem fixa `i' < i` care respectă proprietatea `p ≤ i – i' + 1 ≤ q` şi obţinem trei cazuri:

1. Nu există restricţii în intervalul `[i' + 1, i]`: “reunim” intervalul `[Min[i], Max[i]]` cu `[Min[i'] + 1, Max[i'] + 1]`
2. Există restricţii de o singură valoare (`x`) în intervalul `[i' + 1, i]`: “reunim” intervalul `[Min[i], Max[i]]` cu `{x}`, doar dacă `x - 1` aparţine intervalului `[Min[i'], Max[i']]`
3. Există restricţii de cel puţin două valori distincte în intervalul `[i' + 1, i]`: nu facem nimic

Soluţia are complexitate `O(n * q)` şi, implementată cu grijă, ar trebui să obţină cel puţin 70 de puncte.
Să optimizăm din nou soluţia anterioară. Se observă că cele 3 cazuri reprezintă intervale compacte de valori `i'`. Astfel, putem folosi un arbore de intervale / aib care să de furnizeze răspunsuri de tip minim / maxim pe interval în timp logaritmic şi, astfel, să actualizăm o poziţie a dinamicii în timp logaritmic.

Complexitate finală: `O(n log(n))`

Pentru a calcula minimul lexicografic, pornim cu `OK[n + 1][maxx + 1] = true,` unde, în noile considerente, **`OK[i][j] = true` dacă pe poziţia i poate începe un bloc continuu cu valori egale cu j** (respectiv noile definiţii în acest caz pentru `Max[i]` şi `Min[i]`)**.** Dinamica se construieşte de la dreapta la stânga, într-o manieră similară descrierii de mai sus.

O dată calculată această dinamică, construirea şirului se poate realiza în timp liniar, necesitând totuşi câteva detalii de implementare.

Notăm faptul că există rezolvări care folosesc euristici (greedy) foarte bune pentru a estima valoarea maximă şi pentru a construi şirul, care pot lua punctaje variate.

Solutie 2 (Eugen Nodea):

1) pentru fiecare an fixat reținem cel mai din stânga sector `st[i],` respectiv cel mai din dreapta sector fixat `dr[i]`

2) vom construi doi vectorii auxiliari: `start_st[i]`, dacă se ajunge în sectorul i cu număr minim de ani (p) și `start_dr[i]`, cu număr maxim de ani (`q`). Pentru updatarea următorului sector `i+1` avem cazurile:

```
start_dr[i + 1] = min(start_dr[i + 1], st[i + 1] + MAX - 1)
start_st[i + 1] = max(start_st[i + 1], dr[i + 1] - MAX + 1)
```

3) pentru determinarea numărului maxim de ani vom pleca de la ultimul sector fixat, căutând un ultim an/sector ce permite respectarea condiției necesare (în ultimul an trebuie parcurse cel puțin p sectoare)

4) Pentru construirea soluției minim lexicografic vom pleca de la sfârșit