BARAJ – ZIUA 1 – Descrierea soluţiilor

Excursii

1. Dacă K>P **Nr=0**;
2. Dacă P=K

Vom distribui cei K ghizi în primele K excursii. Rămân N-K persoane care pot fi partiţionate în K clase (cele K excursii disponibile). Deci există $\mathbf{Nr=K^{(N-K)}}$ posibilităţi.

3. Dacă P>K atunci:

Să notăm r=P-K, r≥1.
Repartizăm mai întâi cei K ghizi în K excursii distincte (acestea vor fi excursiile 1, 2, ..., K).
Rămân N-K persoane. Dintre acestea, unele vor fi repartizate în cele P-K=r excursii care nu au ghizi si celelalte vor fi repartizate în excursii care au ghizi.
Să notăm x=numărul de persoane care vor fi repartizate în excursii care nu au ghizi. Acestea pot fi partiţionate în r excursii în S(x,r) moduri (cu S(x,r) notăm numerele lui Stirling de speţa a II-a).
Deci există N-k-x persoane care trebuie repartizate în cele k excursii care au ghizi. Acest lucru se poate realiza în $K^{(N-k-x)}$ moduri.
Cum putem selecta x persoane din cele N-k rămase în C_{n-k}^{x} moduri, rezultă că se obţin:

$$\mathbf{Nr=\sum_{x=r}^{n-k}\left(C_{n-k}^{x}S(x,r)k^{n-k-x}\right)}$$

Numerele lui Stirling de speţa a II-a (S(n,m)) reprezintă numărul de partiţii ale unei mulţimi cu n elemente în m clase (n≥m). Evident, clasele partiţiei trebuie să fie nevide.
Pentru a determina numerele lui Stirling de speţa a II-a se poate utiliza următoare relaţie de recurenţă:
S(n,1)=S(n,n)=1;
S(n,m)=S(n-1,m-1)+m*S(n-1,m).

Robot

Accesibilitate

Regiunea accesibilă unui braţ format dintr-un singur segment de lungime L_1 este un cerc cu raza L_1 şi centrul în origine (umărul robotului).
Regiunea accesibilă unui braţ format din două segmente de lungimi L1, respectiv L2 este un inel, centrat în origine, cu raza exterioară L_1+L_2 şi raza interioară $|L_1-L_2|$.
Aceste rezultate simple pot fi generalizate[1]:
Regiunea de accesibilitate a unui braţ format din n segmente este un inel centrat în origine, cu raza exterioară $ro=L_1+L_2+...+L_n$. Raza interioară este ri=0 (dacă lungimea celui mai lung segment L_{Max} este mai mică decât jumătate din lungimea totală a braţului sau ri=2*LMax-ro.
Acest rezultat ne permite să decidem accesibilitatea în O(n), dar nu ne oferă nici o informaţie despre configuraţia braţului care ne permite accesul la un anumit punct.

[1] Demonstraţia este simplă prin inducţie matematică.

Configuraţia

Braţ format din două segmente

Fie p punctul care trebuie să fie atins. Intersectăm cercul C1 de rază L1, centrat în origine, cu cercul C2, de rază L2, centrat în p.

In general pot exista 0, 1, 2 sau o infinitate de puncte de intersecţie (dacă cele două cercuri coincid). Fie R un punct de intersecţie. Pentru a determina configuraţia braţului, trebuie să calculăm unghiul dintre axa OX şi OR.

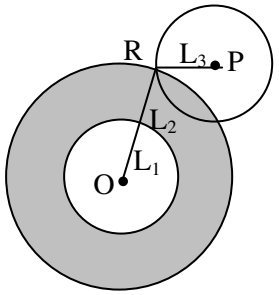
Braţ format din 3 segmente

Vom reduce cazul braţului format din 3 segmente la cazul braţului format din două segmente.

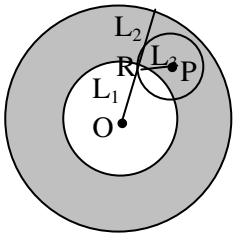
Regiunea de accesibilitate a fragmentului (L_1, L_2) este un inel (să-l notăm A_{12}).

Să examinăm intersecţia dintre cercul C, de rază L3, centrat în p şi inelul A_{12}. Identificăm următoarele situaţii:

1. C intersectează frontiera inelului A_{12}. Fie R un punct de intersecţie. În acest caz problema se poate reduce la cazul braţului format din două segmente aliniind in acelaşi sens (figura a) sau aliniind în sensuri contrare (figura b) L_1 şi L_2.



a.

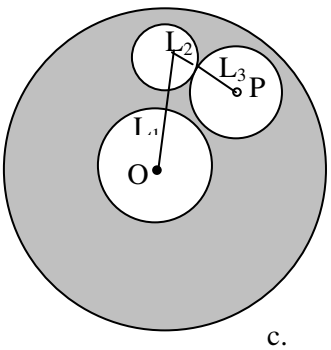


b.

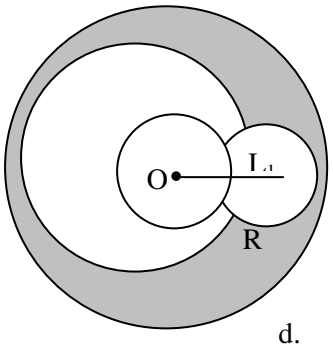
2. C nu intersectează frontierea ineluluiA_{12}.Identificăm în continuare următoarele două subcazuri:

– C nu include originea O (figura c). Fie C2 un cerc de rază L_2 tangent la C. Atunci L_2 şi L_3 pot fi aliniate şi reducem problema la cazul braţului format din două segmente.

– C include originea O (figura d). În acest caz nu există o soluţie cu două segmente aliniate, dar există o soluţie pentru orice valoare a unghiului u_1 (de exemplu u_1=0). Considerăm cercul C1 centrat în extremitatea finală a primului segment P_1. C1 intersectează C în pnctul R. Acest punct ne dă soluţia.



c.



d.

În concluzie orice configuraţie cu 3 segmente se poate reduce la una dintre următoarele situaţii cu 2 segmente:

– L_1+L_2, L_3

– L_1, L_2+L_3

– u_1=0 and L_2, L_3.

Braţ format din n segmente

Dacă un braţ format din n segmente poate atinge un punct, atunci există o configuraţie cu cel mult 2 legături "flexate" (adică cu numai unghiuri diferite de 0). Cele două legături pot fi alese la capetele segmentului median.

Segmentul median este segmentul L_m astfel încât $L_1+L_2+...+L_{m-1}$ <=jumătate din lungimea totală a segmentelor, dar $L_1+L_2+...+L_m$ >decât jumătate din lungimea totală a segmentelor.

Demonstraţie

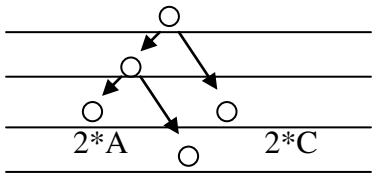
Pentru a demonstra acest rezultat este suficient să moificaţi braţul "îngheţând" toate legăturile, mai puţin cele două indicate. Braţul astfel modificat are aceeaşi regiune de accesibilitate.

Algoritm

Rezultatul precedent ne conduce la un algoritm de complexitate O(n), în care singura parte care depinde de n este calcularea sumei lungimilor segmentelor. După aceasta, aglortimul este de complexitate O(1).

Telegraf

O codificare se poate reprezenta ca un arbore binar, fiecare simbol fiind reprezentat de o frunză. Codificarea simbolului reprezintă drumul de la rădăcină la frunza corespunzătoare; muchia de la un nod la copilul sau stâng este etichetată cu punct, iar muchia spre copilul drept cu linie. Se observă că simbolurile nu se pot găsi decât în frunze (din cauza condiţiei de neambiguitate). Durata transmisiei este dată de suma drumurilor de la rădăcină la fiecare frunză, ponderată cu frecvenţa de apariţie a simbolului; evident o muchie spre copilul drept are lungimea 2, iar o muchie spre copilul stâng lungimea 1. Arborele optim pentru exemplu este:



Liniile orizontale reprezintă adâncimea în arbore. Cum construim un arbore optim? În primul rând, observăm că parcurgerea pe nivele a frunzelor ne dă simbolurile ordonate după frecvenţa apariţiei (simbolurile care apar mai des trebuie să fie în frunzele mai apropiate de rădăcină). În al doilea rând, se observă că arborele trebuie să fie strict (dacă un nod are un singur descendent îl putem elimina şi micşorăm costul).

Folosind aceste observaţii putem construi un algoritm de programare dinamică. Vom construi (conceptual) arborele pe nivele de sus în jos. Următoarele informaţii reprezintă starea:

- A – câte noduri interne sunt pe nivelul L-1
- B – câte noduri interne sunt pe nivelul L-2
- F – câte frunze au apărut deja pe nivelele mai mici decât L

Ce este surpinzător este că nu trebuie să ştim efectiv nivelul L. Vom găsi optimul pentru o anumită configuraţie [A][B][F], la orice nivel s-ar afla. Bineînţeles, dacă nu ştim L nu putem calcula efectiv costul unei frunze când o construim. De aceea costurile trebuie adunate "pe parcurs": când coborâm cu un nivel în arbore, vom aduna costul unui nivel în contul tuturor frunzelor aflate mai jos decât nivelul respectiv. Evident, funzele mai jos de nivelul L au indicii F+1, F+2, … (dacă am ordonat

simbolurile după frecvenţă). Deci, costul unui nivel în plus va fi dat de suma frecveţelor simbolurilor începând cu F+1; aceste sume le vom calcula la începutul algoritmului.

Aşa cum am menţionat, algoritmul construieşte un astfel de array tri-dimensional prin programare dinamică. Memoria folosită este $O(N^3)$, unde am folosit N pentru numărul de simboluri (36 în cazul de faţă). Cel mai simplu mod de a calcula elementele array-ului este o soluţie recursivă cu memoizare. Pentru aceasta, trebuie să vedem din ce poate proveni o soluţie [A][B][F]. Singura variabilă este numărul de frunze de pe nivelul L; pentru X frunze pe nivelul L, starea precedentă este [B-A][A][F-L]. Aceasta ne dă un algoritm în timp $O(N^4)$ pentru constucţia array-ului.

Istoric

Problema este în general studiată pentru cazul generalizat când un punct are costul *a*, iar o linie *b* (la noi a=1, b=2). Pentru *a=b*, Huffman a dat un algoritm în timp O(N), încă în 1951. Despre problema generală (când *a* şi *b* nu sunt constante), nu se cunoaşte un algoritm polinomial, şi nici nu s-a demonstrat ca problema este NP-hard. Cel mai rapid algoritm cunoscut rulează în timp $O(N^b)$ şi a fost descoperit de Bradford, Golin, Larmore şi Rytter în 2000. Soluţia propusă de noi cu timpul de rulare $O(N^{b+2})$ a fost descoperită independent de autorul acestei probleme.

Test	**N**	**Idea**
0	2	nonequal freqs
1	9	equal freqs
2	16	increasing exponentially
3	16	permutation
4	36	100 +/- random(5)
5	36	random(100)
6	36	avg of 3 rand(100)
7	36	1000 +/- random(300)
8	32	1000 +/- random(10)