Backtracking în HEAP

Silviu-Gabriel Udrea

În cadrul acestui articol vom prezenta o modalitate de a îmbunătăți timpul de execuție al programelor care rezolvă probleme folosind metoda backtracking. Un truc relativ simplu ne permite să obținem performanțe de aproximativ trei ori mai bune.

Așa cum știm cu toții, una dintre tehnicile de programare care "consumă" mult timp este *backtracking*-ul.

În zilele noastre, când totul se desfășoară cu o viteză incredibilă, o simplă secundă în plus ne pare a fi o infinitate. Înmulțind această "infinitate" cu numărul de teste pe care suntem nevoiți să le efectuăm pentru a ne verifica algoritmii, timpul obținut astfel devine de-a dreptul insuportabil. Și unde anume avem de efectuat cele mai multe calcule, dacă nu la problemele NP-complete, pe care ne vedem obligați să le rezolvăm prin *backtracking*.

Acest motiv ne determină să acordăm o atenție deosebită optimizării de orice fel, căutând în permanență ceva în plus, mai bun.

În clasa a X-a se studiază operațiile cu pointeri, utilizând liste pentru rezolvarea unor anumite tipuri de probleme. Motivele utilizării *heap*-ului sunt numeroase; nu le vom enumera aici, dar cea mai importantă dintre ele este aceea că la nivelul acestuia operațiile se efectuează mult mai rapid. Așa cum am arătat experimental și cum vă invităm și pe dumneavoastră să o faceți, în *heap* operațiile se efectuează aproximativ de 3,3 ori mai rapid față de cum ar fi fost în cazul alocării statice. Totuși, trebuie precizat faptul că acest raport ține de performanțele calculatorului, deci acesta poate varia de la un sistem de calcul la un altul.

Backtracking-ul ne dă întotdeauna dureri de cap. Pentru a le diminua puțin, vom prezenta o metodă generală de a optimiza de cel puțin trei ori timpul de execuție al programelor care folosesc metoda *backtracking*.

Chiar dacă complexitatea problemei, din punctul de vedere al algoritmului rămâne aceeași, programatorul nu va vedea problema cu aceiași ochi, pentru el "complexitatea" rezolvării fiind "optimizată" de trei ori.

Ideea fiind astfel cristalizată, cum rămâne cu implementarea? După cum vom vedea, implementarea nu presupune un efort deosebit.

Ne întrebăm, firesc, care este motivul pentru care am dori să realizăm o astfel de implementare. Vom da un argument pe care considerăm că îl veți găsi suficient de convingător. O metodă pe care probabil o utilizează mulți la rezolvarea problemelor este aceea de a folosi următorul algoritm:

- se rezolvă problema utilizând metoda *backtracking* și se folosește această rezolvare pentru a obține rezultatele corecte pentru date de intrare mici;
- se caută o anumită particularitate;
- dacă s-a găsit o particularitate, atunci aceasta este studiată și se încearcă rezolvarea problemei fără a folosi metoda *backtracking*.

Un exemplu clar al acestei metode ar putea fi prima problemă de la runda a cincia a *Concursului de Programare Agora*. Se rezolvă problema folosind metoda *backtracking*, se observă modul iterativ prin care se putea genera relația de simetrie (soluția era simetrică) și apoi se implementează soluția corectă.

Deoarece algoritmii care folosesc metoda *backtracking* sunt nestandardizați, vom prezenta doar forma generală a *backtracking*-ului în *heap*.

Prezentăm în continuare un tabel cu date obținute experimental la compararea vitezei generării permutărilor de către cele două tehnici, subliniind faptul că rezultatele sunt mediile a mai multor teste (cel puțin 20):

N	7	8	9	10	11
Back	0,05	0,77	8,25	98,17	1268,90
HeapBack	0,00	0,22	2,47	29,70	384,33

Se observă clar că raportul între timpii de execuție ai programului care folosește metoda *backtracking* sunt de aproximativ 3,3 ori mai mari decât cei ai programului care folosește *backtracking în heap*.

Programul care implementează algoritmul poate fi descărcat de la **www.ginfo.ro/revista/10_6/main.html**.

Silviu-Gabriel Udrea este elev în clasa a XI-a la Colegiul Național "Tudor Vladimirescu" din Târgu-Jiu. Poate fi contactat prin e-mail la adresa **silviu_udrea@yahoo.com**.