Tower8 – descriere solutie

prof.Adrian Panaete – C.N. „A. T. Laurian” Botoşani

La o plasare urmată de operaţiile de upgrade efectul va fi următorul:

1. în poziţia în care se face plasarea se obţine turn de nivel X = cel mai mic nivel care nu este atins în niciunul dintre cei 8 vecini ai poziţiei plasării;
2. în poziţiile vecine toate turnurile de nivel strict mai mic decât X devin poziţii libere.

În aceste condiţii se observă că pentru a ajunge la nivelul X în poziţiile vecine trebuie să avem turnuri de nivel 1,2,..,X-1. De aici se pot trage două concluzii:

1. X-1<=8 deoarece avem cel mult 8 vecini pentru fiecare poziţie deci un turn poate să ajungă cel mult la nivelul 9.
2. Situaţia în care un turn trebuie să ajungă la nivel X şi există printre vecini două turnuri de acelaşi nivel strict mai mic i este neoptimă deoarece s-ar fi putut renunţa la construcţia unuia dintre cele două ajungând la aceeaşi configuraţie într-un număr mai mic de plasări.

Din acest motiv se observă că pentru a ajunge la un turn de nivel X ar fi optim să avem construite în poziţiile vecine exact câte un turn de nivelele 1,2,…,X-1 Deci numărul optim de plasări Nr[x] cuprinde numărul de plasări necesar obţinerii celor X-1 turnuri plus încă o plasare exact înainte de a aplica upgrade până obţinem turnul dorit.
Avem astfel Nr[X] = Nr[1] + Nr[2] + … + Nr[X-1] + 1 cu Nr[1] = 1 Imediat se poate deduce că Nr[1] = 1, Nr[2] = 2, Nr[3] = 4, …, Nr[9] = 256 , mai precis Nr[X] = 2^{X-1}.
Astfel pentru 20 de puncte trebuie adunate valorile Nr[X] pentru absolute toate valorile din matrice.
În continuare mai trebuie să construim un şir de plasări astfel încât să obţinem numărul optim de plasări. Să urmărim cum se ajunge la o plasare optimă a unui turn de nivel X într-o poziţie (i , j) din matrice. Exact înainte de ultima operaţie upgrade vom avea un turn de nivel X-1 pe poziţia dorită şi un turn de nivel X-1 pe una dintre poziţiile vecine dar construit mai devreme. Acest fapt ne sugerează cum se poate construi o funcţie recursivă Pune(i,j,X) care să aibă ca efect construirea unui turn de nivel X la poziţia i,j folosind un număr optim de plasări şi lăsând în rest toate poziţiile din matrice neschimbate.
În funcţie se vor realiza următoarele operaţii :

1. Dacă poziţia este deja ocupată returnează fără niciun efect deoarece orice construcţie trebuie realizată pe o poziţie liberă.
2. Dacă X=1 atunci se plasează pur şi simplu valoarea 1 la poziţia (i,j) şi se returnează pentru că deja scopul funcţiei a fost atins.
3. Dacă X>1 atunci se apelează succesiv funcţia în fiecare vecin cu parametrul X-1 până când apelul are succes. Acest lucru va fi observat când la poziţia vecină în urma apelului se obţine valoarea X-1.
4. Dacă se obţine X-1 la un vecin, apelează Pune(i,j,X-1) având în urma pasului 3 o poziţie vecină ocupată de turn de nivel X-1. Dacă şi acest apel are succes atunci construcţia dorită este realizată şi ne mai rămîne doar să consemnăm acest lucru eliminând turnul de la vecin şi consemnând valoarea X la poziţia (i,j). Dacă însă acest apel nu are succes înseamnă că succesul din apelul de la pasul 3 a fost un fals succes şi în consecinţă vom trece iar la pasul 3 ştergând şi turnul construit la poziţia vecină.

Se observă că funcţia are exact efectul dorit. În plus, se observă că atunci când funcţia returnează la pasul 2. avem exact o plasare care poate fi consemnată într-o stivă de mutări.

Rămâne să stabilim o ordine corectă a apelurilor Pune(i,j,X) unde X este valoarea finală a matricei pe poziţia (i,j).

Observaţia 1.
În orice soluţie optimă turnurile trebuie construite în ordinea descrescătoare a nivelelor.
Demonstraţie
Să considerăm S un şir corect şi minimal de plasşri care rezolvă problema. Pentru orice poziţie din matrice notam Last[i][j] = indicele în S al ultimei plasări efectuate pe poziţia i,j. Acesta este de fapt şi indicele incepând cu care turnul de la poziţia respectivă a ajuns la nivelul său şi nu va mai fi afectat de niciun upgrade efectuat la poziţii vecine. Cum din momentul Last[i][j] pozitia(i,j) nu va mai fi afectata de poziţiile vecine, deducem ca în poziţiile vecine orice turn are nivel strict mai mare trebie deja să fie construit (altfel upgrade-ul acelui turn ar utiliza turnul curent măcar la ultimul său upgrade). Deci dacă avem doua poziţii vecine (i1,j1) şi (i2,j2) cu nivel[i1][j1]>nivel[i2][j2] atunci last[i1][j1]<last[i2][j2]. De aici deducem că pentru orice turn în momentul plasării care conduce la forma sa finală turnurile vecine de nivel mai mare trebuie sa fie deja construite.

Conform acestei observaţii apelurile vor fi facute începând cu turnurile de nivel 9 şi terminând cu cele de nivel 1.

Rămâne acum să vedem în ce ordine vom plasa turnurile de acelaşi nivel. O idee ar fi să le plasăm la intâmplare sau eventual parcurgând matricea după anumite reguli fixe (de exemplu de la stânga la dreapta şi de sus în jos). Din păcate această abordare

care dă rezultate bune pentru teste random nu rezolvă problema în cazul general deoarece la anumite configuraţii construirea unui turn de un anumit nivel va face imposibilă contrucţia ulterioară a altui turn de acelaşi nivel.
O astfel de abordare ar aduce în mod normal cel mult 50% din punctaj.

Soluţia optimă trebuie să ţină cont că la orice moment există cel puţin un turn care poate fi construit. Vom procesa turnurile pe rând incepând cu cele de nivel 1 în ordine crescătoare şi cu matricea în forma finală. Când procesăm turnurile de un anumit nivel punem toate poziţiile corespunzătoare într-o coadă şi procedăm astfel: ștergem turnul de pe matrice si verificăm dacă apelul *Pune* ne duce cu succes la construcţia turnului dorit atunci scoatem poziţia din coadă şi îl punem într-o stivă înţelegând astfel că acel turn va fi construit după ce se constriuesc toate turnurile rămase în coadă. Dacă nu, mutăm poziţia la finalul cozii pentru că acel turn va fi construit mai devreme folosind poziţii ale unor alte turnuri din coadă care vor fi plasate mai repede în stivă deci care se vor construi mai târziu. În această etapă nu este necesar ca odată cu apelul funcţiei recursive să se consemneze şi plasările efectuate. După această procesare stiva va conţine toate poziţiile din matrice de la nivel mare în vârful stivei spre nivel mic la baza stivei, exact în ordinea în care ele trebuie construite pentru a obţine o soluţie optimă. Acum mai rămâne să înepem apelurile exact în ordinea indicată de stivă, consemnând de data aceasta şi plasările efectuate de fiecare apel realizat cu succes.