SORTAREA PRIN SELECŢIE ARBORESCENTĂ

PREZENTAREA METODEI

Principiile sortării prin selecție arborescentă sunt uşor de înțeles gândindu-ne la un turneu cu meciuri eliminatorii a unei competiții de tenis.

Dacă sunt 8 jucători (A, B, C, D, E, F, G şi H) şi rezultatele ar fi cele din figura de mai jos. La nivelul de la bază, jucătorul ***A*** îl învinge pe jucătorul ***B*** şi jucătorul ***D*** îl învinge pe jucătorul ***C***, după care în turneul următor jucătorul ***D*** îl învinge pe jucătorul ***A*** etc.

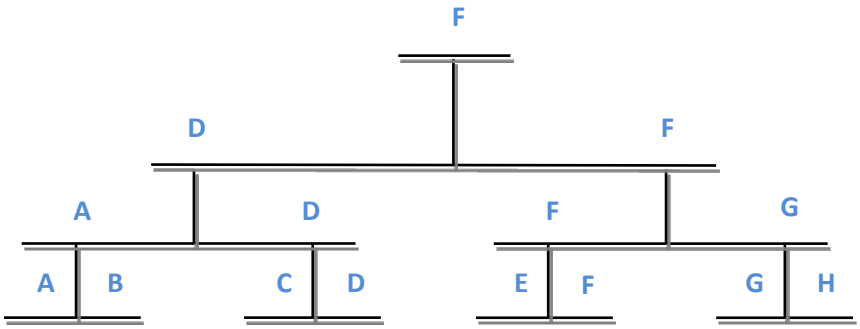


Figura 5. Tabloul meciurilor de la o competiție de tenis

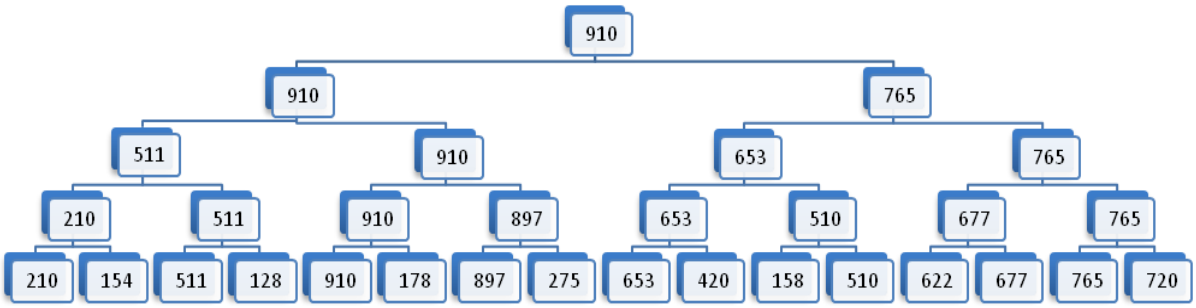
Campionul este ***F*** dintre cei 8 jucători, iar pentru a stabili acest lucru au fost necesare 7 meciuri. Jucătorul ***D***, nu este neapărat cel mai bun jucător după ***F***, deoarece oricare dintre jucătorii învinşi de jucătorul ***F***, poate fi mai bun decât jucătorul ***D***. Se poate determina vicecampionul prin rezultatul meciului dintre jucătorii ***E*** şi ***G*** iar câştigătorul acestui meci se va confrunta cu jucătorul ***D***. Deci sunt necesare doar două meciuri pentru a găsi ocupantul locului doi.

La turneu au participat 8 jucători şi au fost necesare 7 (=8-1) meciuri (în cazul nostru comparații), pentru a determina câştigătorul. În general se poate scoate jucătorul din rădăcina arborelui şi cel al doilea jucător bun va apare în rădăcina arborelui, dacă se recalculează învingătorii din nivelele superioare ale arborelui. În acest scop trebuie schimbată, numai o parte a arborelui astfel că sunt necesare mai puțin de [***log n***] comparații suplimentare pentru selectarea următorului jucător bun. Tot în acest mod se va găsi cel de-al treilea jucător etc. Timpul total pentru o astfel de sortare prin selecție este direct proporțional cu [***n log n***].

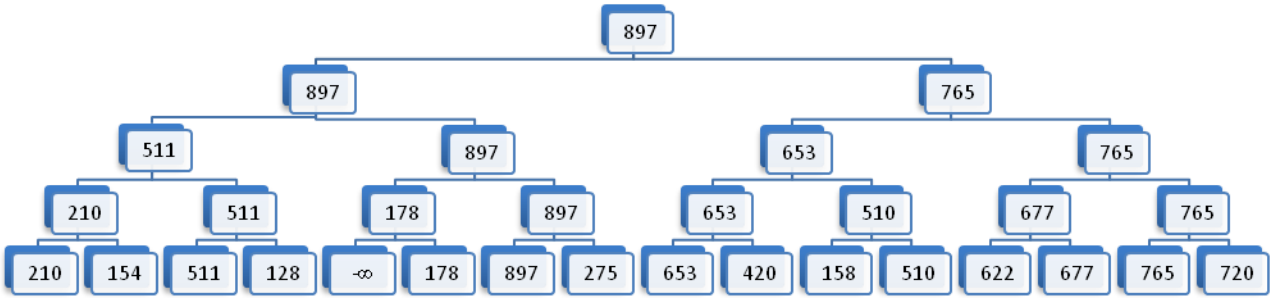
Vom arata sortare prin selecție arborescentă pentru un vector cu 16 numere (210, 154, 511, 128, 910, 178, 897, 275, 653, 420, 158, 510, 622, 677, 765, 720). Este necesar să se cunoască de unde vine

cheia în rădăcina, în scopul cunoaşterii locului în care se introduce "−∞" următor. Astfel, nodurile de ramificare a arborelui actual, conțin un indicator sau index, care specifică poziția cheii semnificative, în locul elementului însuşi.

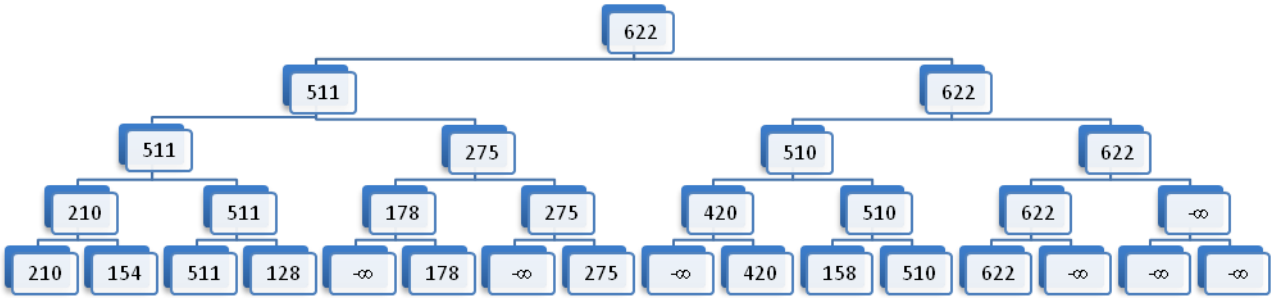
Figura 1. Exemplu de sortarea prin selecție arborescentă



(a) Configurația inițială



(b) 910 este înlocuit cu simbolul -∞ şi următorul element în ordinea mărimii se deplasează în sus către rădăcină



(c) ***Configurația după ce 910, 897, 765, 720, 677, 653 au fost scoase ca rezultate de ieşire***

Pentru a simplifica selecția arborescentă, K.. E. Iverson a renunțat la indicatori anticipând în modul următor: când câştigătorul unui meci de pe nivelul de la bază a arborelui este deplasat mai sus, el poate fi înlocuit imediat cu "-∞" pe nivelul de bază. Când câştigătorul se deplasează mai sus de pe o

ramură pe alta, putem să-1 înlocuim cu acela care eventual, ar fi fost deplasat mai sus în locul eliberat (adică cheia mai mare din cele două chei de dedesubt).

Repetând această operație obținem următoarea structură arborescentă.

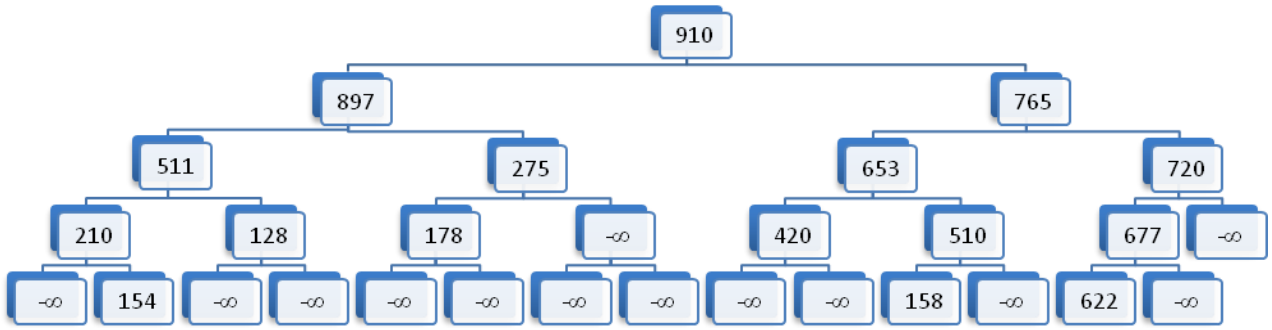


Figura 2. Principiul lui Peter la sortare. Fiecare element se ridică în ierarhie până la nivelul său de competență.

Odată ce arborele a fost realizat în această formă, putem trece la sortare prin metoda " vârf - bază " în locul metodei " bază - vârf ". Se scoate rădăcina, se mişcă în sus descendentul cel mai mare, se mişcă în sus următorul descendent cel mai mare, etc. Acest procedeu arată ca un sistem de promovare în cadrul unei companii.

Se observă că metoda " vârf - bază " are avantajul că pot fi evitate comparațiile redundante pentru "− ∞" cu "− ∞". La metoda "de jos în sus" se tot găseşte "− ∞" în ultimele stadii ale sortării, dar abordarea prin metoda " vârf - bază " poate opri modificarea arborelui pe parcursul fiecărui stadiu, de câte ori a fost sortat "− ∞".

Întrebarea care s-ar pune acum ar fi dacă n-am putea folosi metoda de mai sus fără să recurgem deloc la − ∞. Răspunsul este că putem nu numai elimina − ∞, dar putem să evităm necesitatea unei zone de ieşire auxiliare. Acest raționament ne conduce la un important algoritm de sortare numit ***sortare heap (heapsort)*** realizat de J. W. J. Williams.