Căutarea informată

Căutarea informată se mai numeşte şi *căutare euristică*. Euristica este o metodă de studiu şi de cercetare bazată pe descoperirea de fapte noi. În acest tip de căutare vom folosi informaţia despre spaţiul de stări. Se folosesc cunoştinţe specifice problemei şi se rezolvă probleme de optim.

Căutarea de tip best-first

Tipul de căutare pe care îl discutăm aici se aseamănă cu tehnica breadth-first, numai că procesul nu se desfăşoară în mod uniform plecând de la nodul iniţial. El înaintează în mod preferenţial de-a lungul unor noduri pe care informaţia euristică, specifică problemei, le indică ca aflându-se pe drumul cel mai bun către un scop. Un asemenea proces de căutare se numeşte *căutare euristică* sau *căutare de tip best-first*.

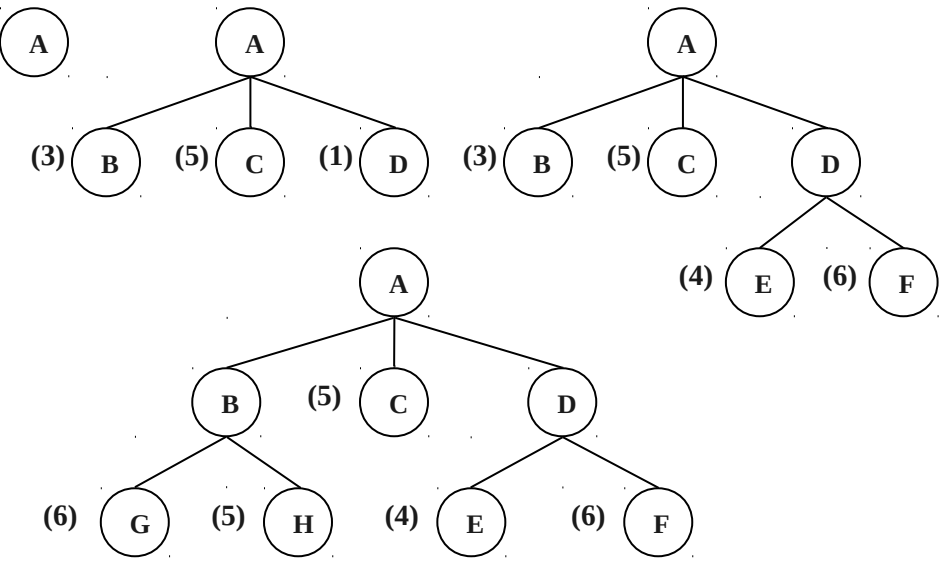
Principiile pe care se bazează căutarea de tip best-first sunt următoarele:

1. Se presupune existenţa unei funcţii euristice de evaluare, $\hat{f}$, cu rolul de a ne ajuta să decidem care nod ar trebui extins la pasul următor. Se va adopta *convenţia* că valori mici ale lui $\hat{f}$ indică nodurile cele mai bune. Această funcţie se bazează pe informaţie specifică domeniului pentru care s-a formulat problema. Este o funcţie de descriere a stărilor, cu valori reale.

2. Se extinde nodul cu cea mai mică valoare a lui $\hat{f}(n)$. *În cele ce urmează, se va presupune că extinderea unui nod va produce toţi succesorii acelui nod.*

3. Procesul se încheie atunci când următorul nod care ar trebui extins este un nod-scop.

Fig. 2.7 ilustrează începutul unei căutări de tip best-first:



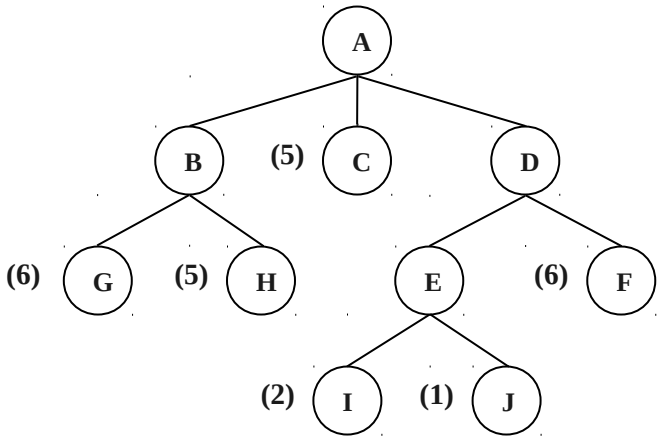


Fig. 2.7

În Fig. 2.7 există iniţial un singur nod, A, astfel încât acesta va fi extins. Extinderea lui generează trei noduri noi. Funcţia euristică este aplicată fiecăruia dintre acestea. Întrucât nodul D este cel mai promiţător, el este următorul nod extins. Extinderea lui va produce două noduri succesor, E şi F, cărora li se aplică funcţia euristică. În acest moment, un alt drum, şi anume acela care trece prin nodul B, pare mai promiţător, astfel încât el este urmat, generându-se nodurile G şi H. În urma evaluării, aceste noduri par insă mai puţin promiţătoare decât un alt drum, astfel încât este ales, în continuare, drumul prin D la E. E este apoi extins producând nodurile I şi J. La pasul următor va fi extins nodul J, întrucât acesta este cel mai promiţător. Procesul continuă până când este găsită o soluţie.

Pentru a nu fi induşi în eroare de o euristică extrem de optimistă, este necesar să înclinăm căutarea în favoarea posibilităţii de a ne întoarce înapoi, cu scopul de a explora drumuri găsite mai devreme. De aceea, vom adăuga lui $\hat{f}$ *un factor de adâncime*: $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$, unde $\hat{g}(n)$ este *o estimaţie a adâncimii lui n în graf,* adică reprezintă lungimea celui mai scurt drum de la nodul de start la *n*, iar $\hat{h}(n)$ este *o evaluare euristică a nodului n.*

Pentru a studia, în cele ce urmează, *aspectele formale ale căutării de tip best-first*, vom începe prin a prezenta un algoritm de căutare generală bazat pe grafuri. Algoritmul include versiuni ale căutării de tip best-first ca reprezentând cazuri particulare.

Algoritm de căutare general bazat pe grafuri

Acest algoritm, pe care îl vom numi GraphSearch, este unul general, care permite orice tip de ordonare preferată de utilizator - euristică sau neinformată. Iată o *primă variantă* a definiţiei sale:

GraphSearch

1. Creează un arbore de căutare, T_r, care constă numai din nodul de start n_0. Plasează pe n_0 într-o listă ordonată numită OPEN.

2. Creează o listă numită CLOSED, care iniţial este vidă.

3. Dacă lista OPEN este vidă, EXIT cu eşec.

4. Selectează primul nod din OPEN, înlătură-l din lista OPEN şi include-l în lista CLOSED. Numeşte acest nod *n*.

5. Dacă *n* este un nod scop, algoritmul se încheie cu succes, iar soluţia este cea obţinută prin urmarea în sens invers a unui drum de-a lungul arcelor din arborele T_r, de la *n* la n_0. (Arcele sunt create la pasul 6).

6. Extinde nodul *n*, generând o mulţime, *M*, de succesori. Include *M* ca succesori ai lui *n* în T_r, prin crearea de arce de la *n* la fiecare membru al mulţimii *M*.

7. Reordonează lista OPEN, fie în concordanţă cu un plan arbitrar, fie în mod euristic.

8. Mergi la pasul 3.

❑

4. Observaţie: Acest algoritm poate fi folosit pentru a efectua căutări de tip best-first, breadth-first sau depth-first. În cazul algoritmului *breadth-first* noile noduri sunt puse la sfârşitul listei OPEN (organizată ca o coadă), iar nodurile nu sunt reordonate. În cazul căutării de tip *depth-first* noile noduri sunt plasate la începutul listei OPEN (organizată ca o stivă). În cazul căutării de tip *best-first*, numită şi căutare euristică, lista OPEN este reordonată în funcţie de meritele euristice ale nodurilor.