Formulazione del problema prevede lo stato iniziale, l’insieme degli operatori che abbiamo a disposizione, test obiettivo (una proprietà), funzione di costo cammino.

**Idea**: mantenere ed estendere sequenze che sono soluzioni parziali. Viene costruito uno spazio di ricerca con un Albero di ricerca. L’esplorazione coincide con verificare che uno stato corrisponde al goal oppure l’agente determina gli stati raggiungibili da questo.

**Espansione**: Partire da uno stato iniziale e applicando operatori generare stati

**Strategia di ricerca**: Ad ogni passo scegliere quale nodo espandere tra quelli ottenuti.

**Albero di ricerca**: L’espansione degli stati a partire da quello iniziale.

**Strategie non informate**: Non applicano conoscenza sul problema:

1. **Breadth First**

Viene gestita una coda Fifo. Con branching B e profondità D, Tempo = Spazio = BD.

Svantaggio: Complessità spaziale esponenziale, Ma completa. trova sempre la soluzione.

1. **Depth First**

Viene gestita una coda Lifo. Tempo = BM, Spazio = bm.

Può essere non completa, se vi è un loop.

1. **Depth First a profondità limitata**

Quando si raggiunge il massimo di profondità o vi è un fallimento o si considerano strade alternative. Evita di scendere lungo rami infiniti.

1. **Iterative Deepening**

È una #3, ma con limite c. Se si trova la soluzione termina. Altrimenti, incrementa c++.

Completa, complessità temporale: BD, spaziale: bd.

**Strategie informate**: Invece di espandere i nodi in modo qualunque. usiamo conoscenza euristica sul problema.

**Funzioni di valutazione**: Danno una stima computazionale dello sforzo per raggiungere lo stato finale.

1. **Best First**

Usa una funzione di valutazione che calcola un numero che rappresenta la desiderabilità relativa all’espansione del nodo. **Sceglie il nodo più desiderabile**. I **successori** vengono inseriti **in ordine decrescente** di desiderabilità, si **cerca di muoversi verso il minimo** di una funzione che stima il costo per raggiungere il goal.

Casi particolari:

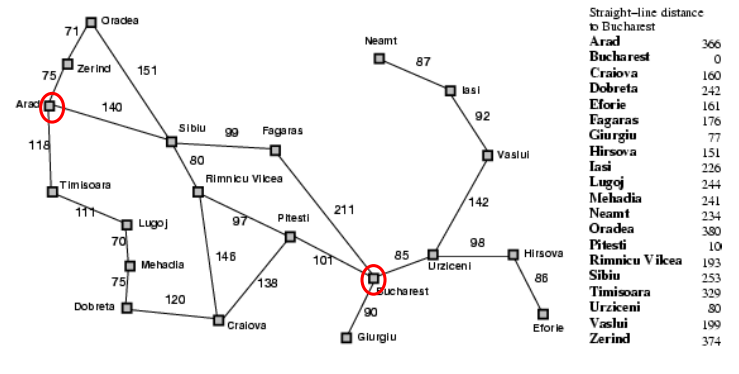
I.a **Ricerca Greedy**

**Algoritmo**:

Sia nodes, lista dei nodi del problema, ordinati secondo la loro distanza dal goal (crescente)

* Se nodes è vuota fallisci; altrimenti sia N il primo nodo di nodes.
* Se N è il goal fermati e restituiscilo insieme alla strada percorsa per raggiungerlo
* Altrimenti rimuovi N da nodes
  + Aggiungi a nodes tutti i nodi figli di N con label la strada percorsa partendo dal nodo iniziale.
  + Poi ordina tutta la lista nodes in base alla stima della distanza relativa dal goal.
  + Ritorna al passo 2

Non è ottimale e può essere incompleta. Nel caso peggiore, la complessità temporale è BD e anche quella spaziale.



I.b **Ricerca A**

Invece di considerare solo la distanza dal goal. Considera **anche il costo nel raggiungere il nodo N dalla radice**.

Espandiamo quindi i nodi in ordine crescente di f(n) = g(n) + h'(n) con g(n) è la profondità del nodo (o il costo del cammino, nel caso di costo non unitario degli operatori) e h'(n) la distanza dal goal.

**Algoritmo**:

Sia nodes una lista dei nodi iniziali (non espansi) del problema.

* Se nodes è vuota fallisci; altrimenti, sia N il nodo di nodes per cui g(n) + h'(n) è minima.
* Se N è il goal fermati e restituiscilo insieme alla strada percorsa per raggiungerlo
* Altrimenti rimuovi N da nodes e aggiungi a nodes tutti i nodi figli di N con label la strada percorsa partendo dal nodo iniziale.
* Ritorna al passo 2.