Algoritmos Paralelos de Busca em Grafos

- Algoritmos desenvolvidos para resolver problemas na teoria de grafos
  - Busca em grafos;
  - Encontrar uma solução válida.
- Força Bruta
  - Dividir para Conquistar
  - Busca em Largura
  - Busca em Profundidade
- Heurística
  - Branch & Bound, A*
- Adversary Search
  - Minimax, Alpha-beta

Dividir para Conquistar

- Metologia
  - Particionar um problema em subproblemas
  - Resolver os subproblemas
  - Combinar as soluções dos subproblemas
- Recursividade
  - Subproblemas podem ser solucionados utilizando a metodologia Dividir&Conquistar
- Busca desordenada

Dividir para Conquistar

- Arquitetura Multiprocessada Centralizada (multicore)
  - Subproblemas não resolvidos mantidos em uma pilha
  - Processadores ociosos podem acessar a pilha
  - Processadores com cobrecarga de trabalho podem empilhar subproblemas.
  - Mecanismo efetivo de balanceamento de trabalho
  - A pilha pode se tornar um gargalo com o aumento do número de processadores

Dividir para Conquistar

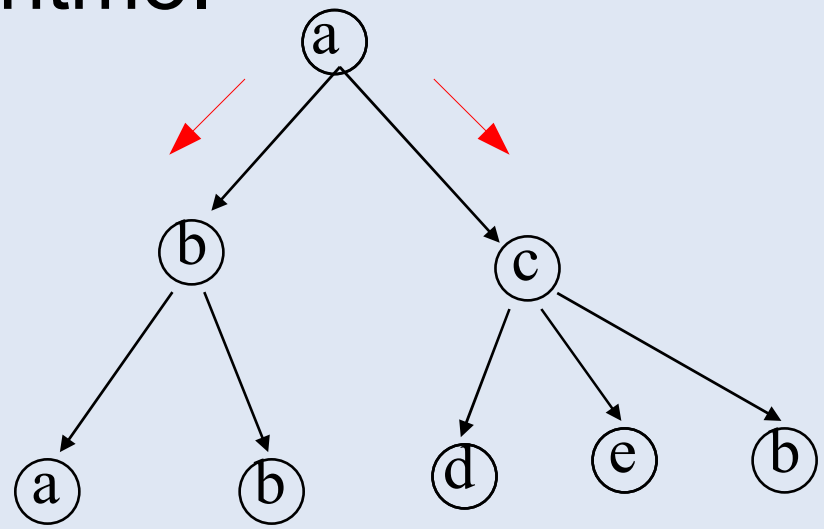
- Arquitetura de multicomputadores (*clusters*)
  - Subproblemas são distribuídos entre processadores individuais
  - Problema original e solução final são armazenados na memória de um único processador
    - O problema é dividido e propagado
    - Processadores computam as soluções para os subproblemas
    - Resultados parciais são combinados

Busca em Profundidade

- Explora completamente cada *ramo* da árvore antes de verificar o ramo vizinho.
- Sempre expande primeiro o nó no nível mais profundo da árvore.
- Mantém uma estrutura de *fila* para armazenar os nós ainda não expandidos.
  - Os nós são adicionados no **início** da fila.
- Não é completa e não é ótima.

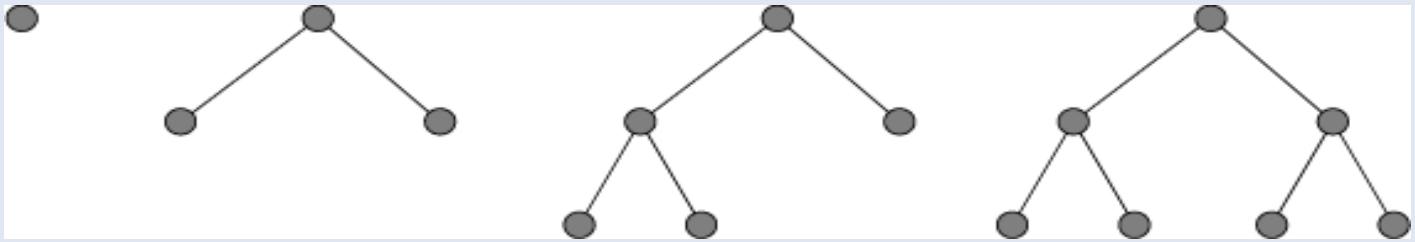
Busca em Profundidade

- Arquitetura paralela
  - Distribuir os nós não expandidos e retornar o resultado da expansão
- Alto sincronismo para garantir a integridade do algoritmo.



Busca em Largura

- Explora completamente cada *nível* da árvore antes de verificar o nível seguinte.
- Expande todos os nós nível por nível.
- Mantém uma estrutura de *fila* para armazenar os nós ainda não expandidos.
  - Os nós são adicionados no **final** da fila.
- É completa e ótima.



Busca em Largura

- Arquitetura paralela
  - Distribuir os nós não expandidos nível a nível e retornar o resultado da expansão
- Baixo sincronismo.

Best-First

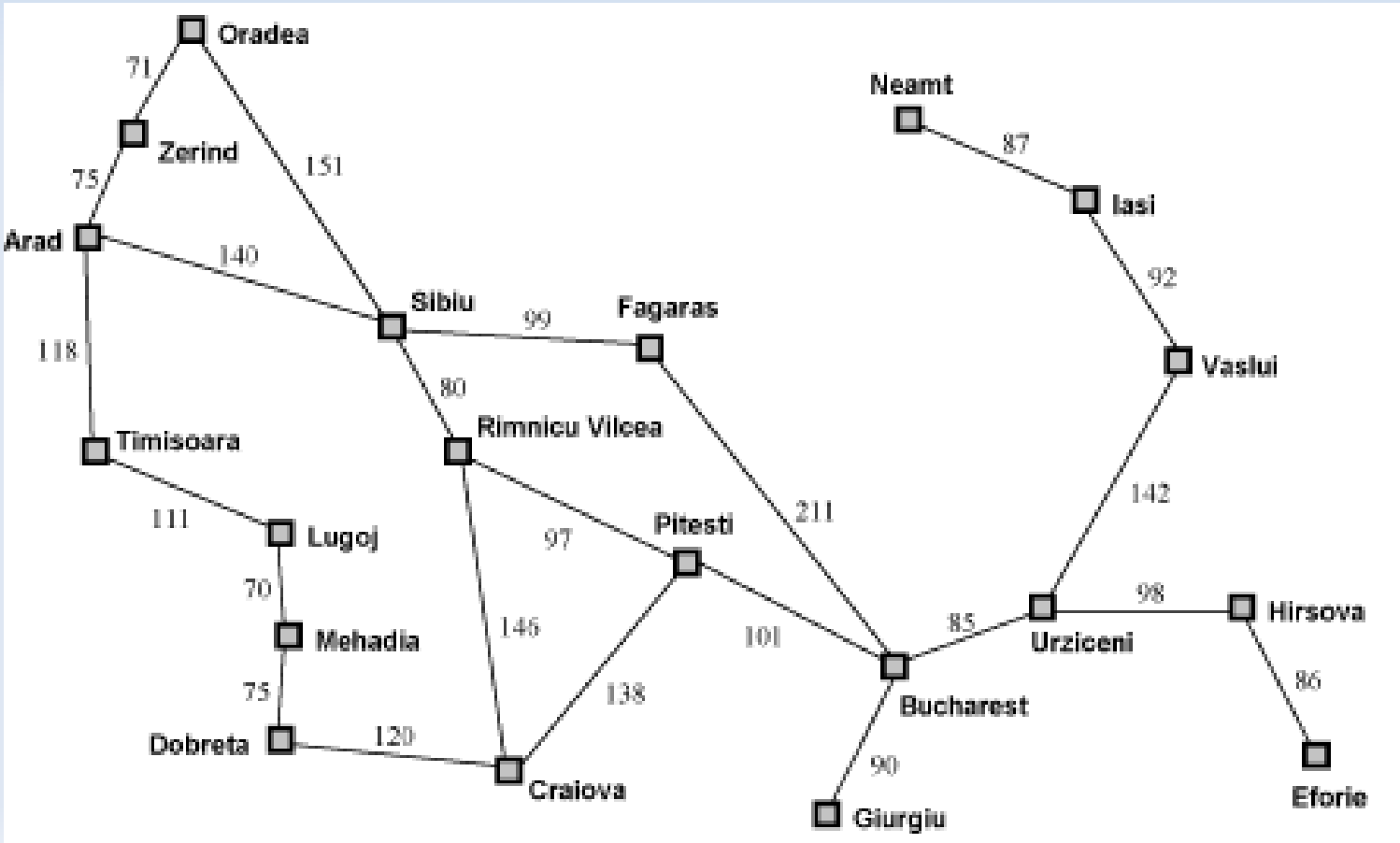
- Busca genérica onde o **nó de menor custo "aparente"** na fronteira do espaço de estados é expandido primeiro.
- Duas abordagens básicas:
  - 1. Busca Gulosa (*Greedy Search*)
  - 2. Algoritmo A*
- Busca gulosa
  - Semelhante à **busca em profundidade**
  - Tenta expandir o nó mais próximo ao nó final com base na estimativa feita pela função heurística ***h***.

Best-First

- Arquitetura paralela
  - Expandir os vértices em paralelo
    - A cada instante de tempo, *n* melhores vértices são considerados para expansão.
    - Ao final da expansão cada processador coloca os vértices gerados na lista.

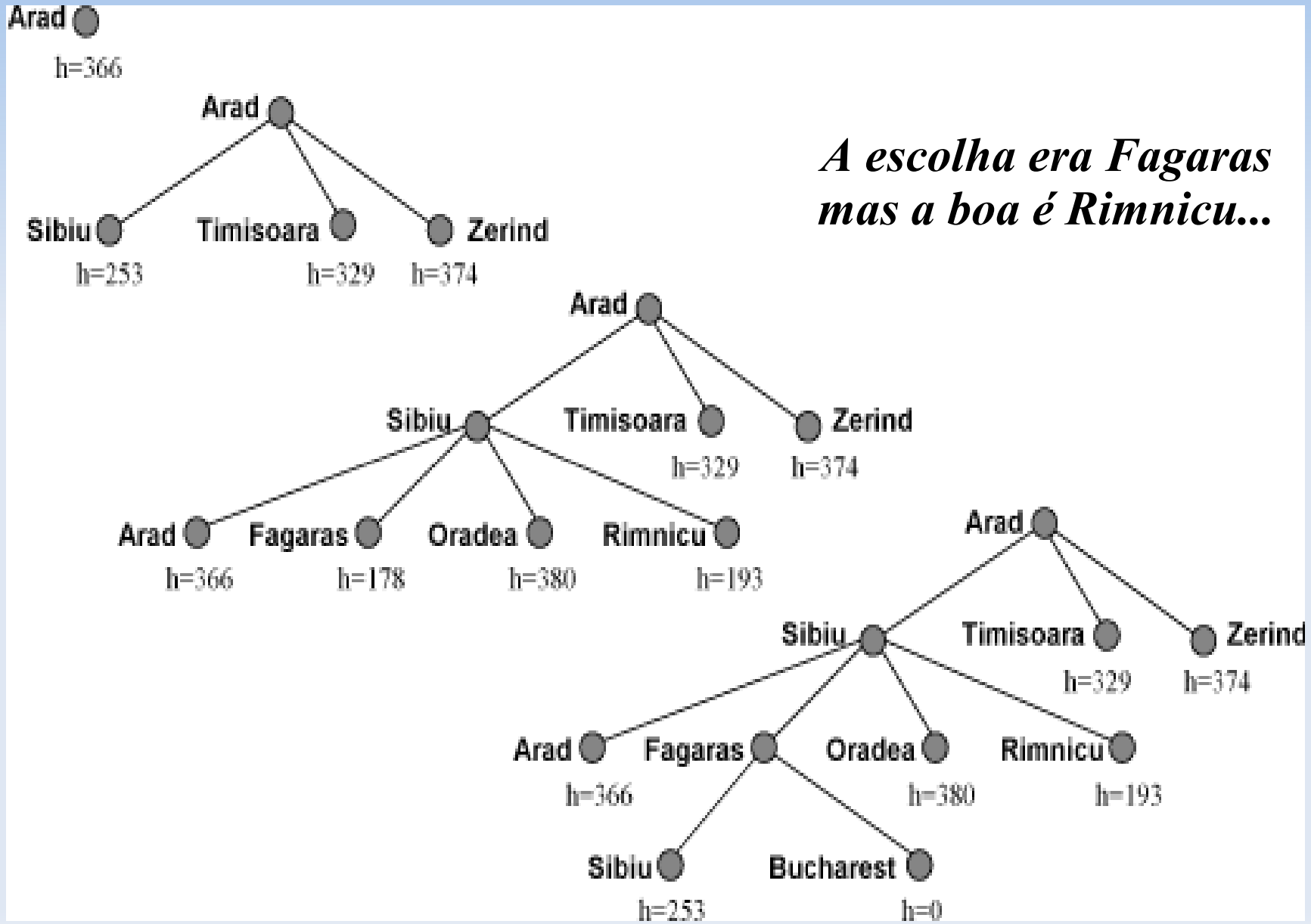
Busca Gulosa

- Exemplo:
  - encontrar a rota mais curta de Arad a Bucharest
  - hdd(n) = distância direta entre o nó ***n*** e o nó final



Straight line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Busca Gulosa



A escolha era Fagaras mas a boa é Rimnicu...

Busca Gulosa

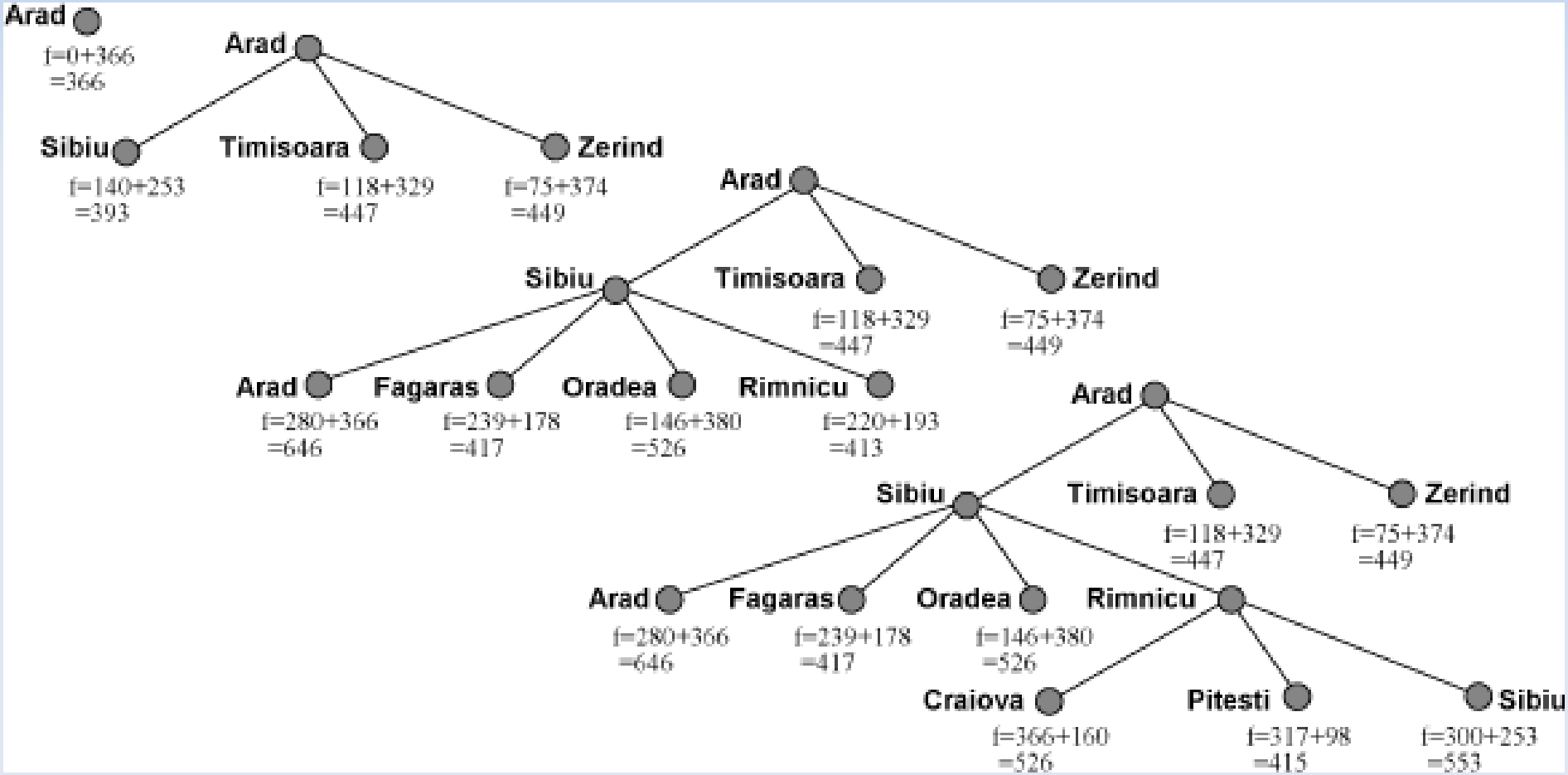
- Não é ótima... **(semelhante à busca em profundidade)**
  - porque só olha para o futuro!
- Não é completa:
  - pode entrar em "*loop*" se não detectar a expansão de estados repetidos
  - pode tentar desenvolver um caminho infinito
- Custo de busca é minimizado
  - não expande nós fora do caminho
- Porém... escolhe o caminho que é mais econômico à primeira vista
- E no entanto, podem existir caminhos mais curtos...

Algoritmo A*

- Tenta minimizar o custo total da solução combinando:
  - Busca Gulosa
    - econômica, porém não é completa nem ótima
  - Busca de Custo Uniforme
    - ineficiente, porém completa e ótima
- Função de avaliação:
  - f(n) = g(n) + h(n)
  - g(n) = distância de ***n*** ao nó inicial
  - h(n) = distância estimada de ***n*** ao nó final
- A* expande o nó de menor valor de ***f*** na fronteira do espaço de estados.
  - Olha o futuro sem esquecer do passado!

Algoritmo A*

- Qual seria a rota desenvolvida pelo A* entre as cidades de Arad e Bucharest?



Algoritmo A*

- A* completo e ótimo
  - como a busca em largura...
- A* é otimamente eficiente:
  - não existem algoritmos que expandem menos nós com a mesma *f*

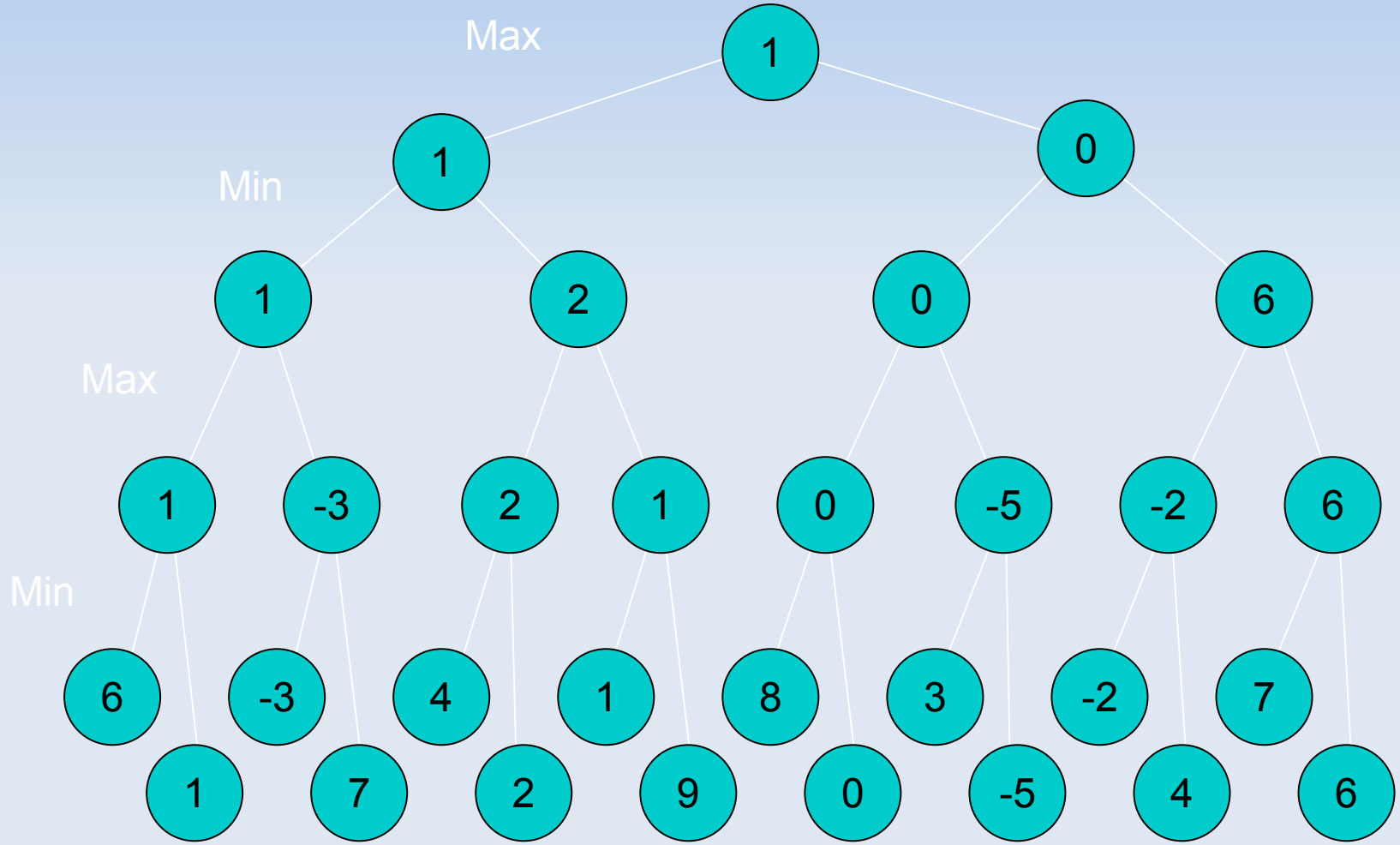
Branch&Bound

- Branch:
  - Utiliza algum algoritmo de busca em árvore já visto
    - Largura, profundidade, best-first
- Bound:
  - Utiliza informação sobre a otimalidade de soluções paciais para podar/limitar a árvore de busca.
- Abordagem paralela
  - Master-Slave

Minimax

- Melhores algoritmos para aplicações em jogos de tabuleiro (xadrez, damas, jogo-da-velha, ...)
- O algoritmo considera todos os movimentos possíveis em *n* níveis de uma árvore, avalia o resultado dos movimentos e propaga os resultados para dentro da árvore de busca para determinar qual o melhor movimento inicial.
- Essencialmente é um algoritmo *depth-first branch&bound*.

Minimax



Minimax

- Arquiteturas paralelas:
  - Paralelizar a geração de movimentos e a avaliação das posições
  - Paralelizar a busca na árvore. Criar sub-árvores independentes.
    - IBM`s Deep Blue