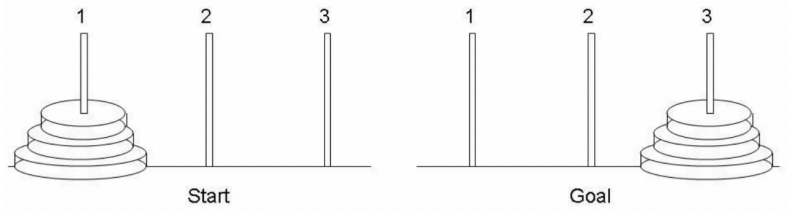
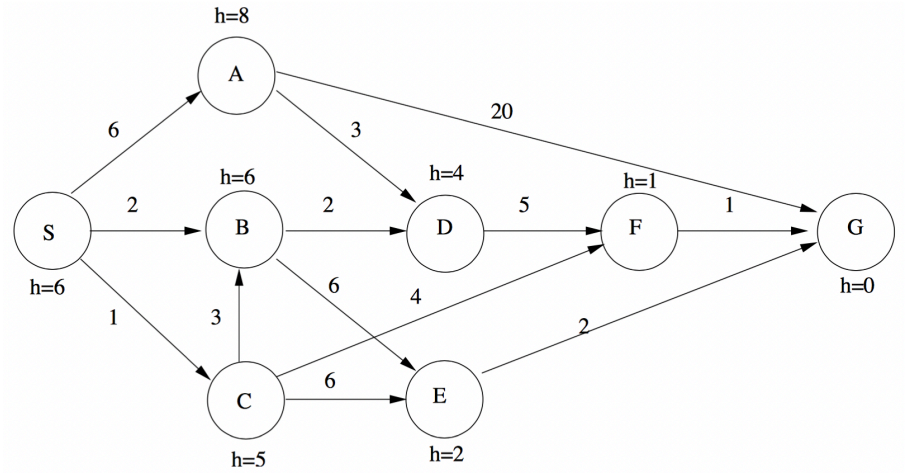
1. A Torre de Hanoi é um jogo onde uma pilha de discos de variados tamanhos deve ser movida de uma haste para outra. A figura abaixo mostra a posição inicial do jogo e a posição final para a versão com três discos e três hastes.



As regras do jogo são de que um disco só pode ser movido se ele estiver no topo de uma pilha e só pode ser colocado em uma haste vazia ou sobre um disco maior. Vamos chamar de S, o disco menor, com peso de 1u, M, o mediano, com peso de 2u e L, o maior, pesando 3u. O custo de um movimento é o peso do disco multiplicado pela distância do movimento (1 para uma haste adjacente e 2 para uma não adjacente). Formule este problema como um problema de busca, priorizando uma representação já mais próxima da computacional, e ilustre um pedaço da árvore de espaço de estados.

2. Considere o grafo abaixo, que representa a formulação de um problema de busca global, com estado inicial S e estado final G. Os custos reais encontram-se nos arcos e o valor de heurística especificado pela letra h. Pede-se:



(a) Qual o caminho-solução encontrado pelo A*?

(b) A heurística deste problema é admissível? Justifique.

(c) A heurística deste problema é consistente (ou monotônica)? Justifique.

3. Suponha que h1 e h2 são heurísticas admissíveis. Responda:

(a) Circule a(s) heurística(s) seguinte(s) que também é(são) necessariamente admissível(is)?

h3 = h1+(h2)/2, h4 = (h1+h2)/2, h5 = 2*h1, h6 = min(h1,h2), h7 = h1*h2/10

(b) Qual das seguintes heurísticas é a melhor: h8 = max(h1, h2) ou h9 = (h1+h2)/2? Explique.

4. Qual será o comportamento do algoritmo A* se h(n) é uma heurística perfeita que sempre retorna o custo exato para o estado objetivo?

5. O RobUFS deve navegar entre os prédios do campus de São Cristóvão para entregar correio. Ele implementa o algoritmo A* para planejamento de percurso (usando distância em linha reta como função heurística), mas ele não está trabalhando corretamente uma vez que percorre com frequência um caminho sub ótimo. Abaixo segue o pseudocódigo que ele usa e que foi implementado por alunos de IA do DCOMP. Qual é o erro no pseudocódigo e por que ele leva a soluções sub ótimas?

```
inicialize: let Q = {S}
enquanto Q <> {}
       retire Q1    //1o elemento de Q
       nós-filhos = expand(Q1)
       elimine nós-filhos que representem loops
```

```
        para cada filho restante em nós-filhos faça
              se filho é um estado-objetivo então FIM
        adicione filhos do nó em Q
        ordene Q de acordo com fcost = custo(nó)+h(nó)
FIM
```

6. No grafo abaixo, onde S é o nó inicial e F e G são os nós objetivo, cada arco contém o custo para atravessá-lo e dentro de cada nó existe uma estimativa do custo até o nó objetivo mais próximo. Utilizando a estratégia de busca A*,

(a) desenhe a árvore de busca;

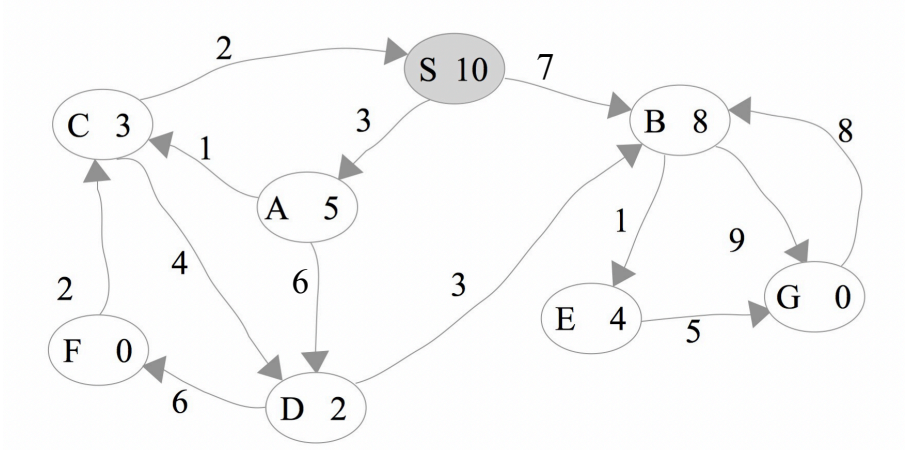
(b) mostre a lista de nós expandidos;

(c) circule na árvore a solução encontrada (caso exista) e seu respectivo custo;

(d) mostre a fronteira final;

(e) a solução é ótima? Justifique.

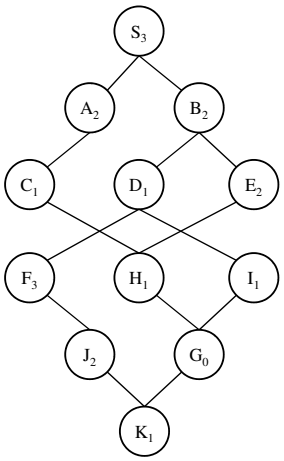
(OBS: Na ausência de outro critério, ordene os nós em ordem alfabética. Efetue a solução como busca em árvore. Em caso de empate, expanda o primeiro nó gerado)



.

7. Considere o problema de mover o cavalo em um tabuleiro 3x4, com estados inicial S e final G, conforme figura abaixo. O espaço de busca pode ser representado no grafo ao lado do tabuleiro. O dígito subscrito em cada nó é o valor da heurística. Todas as transições possuem custo 1.

S_3	H_1	D_1	K_1
I_1	J_2	A_2	E_2
C_1	B_2	G_0	F_3



Escreva a sequência de nós na ordem visitada pelos métodos especificados. Você pode achar útil desenhar a árvore de busca correspondente ao grafo acima. Faça as seguintes suposições: 1. Os algoritmos não geram caminhos com loops. 2. Os nós são selecionados em ordem alfabética quando o algoritmo encontra um empate. 3. Um nó é visitado quando esse nó está na frente da fila de fronteira.

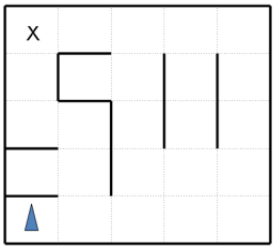
(a) depth-first search.

(b) breadth-first search.

(c) A* search (além da sequencia de nós visitados, informe o custo total estimado f(s)=g(s)+h(s) para cada nó visitado e retorne o

caminho final encontrado)

8. Imagine que um carro autônomo deseje sair de um labirinto como o mostrado ao lado. O agente é direcional, com direções d ∈ (N, S, E, W). Em uma única ação, o agente pode avançar a uma velocidade ajustável v ou girar. As ações de girada são à esquerda e à direita, que mudam a direção do agente em 90 graus. O giro só é permitido quando a velocidade é zero (e continua sendo zero após o giro). As ações possíveis em movimento são Rápido e Lentao. Rápido incrementa a velocidade em 1 e Lento diminui a velocidade em 1; em ambos os casos, o agente move um número de quadrados igual à sua NOVA velocidade ajustada. Qualquer ação que resulte em uma colisão com uma parede trava o agente e é ilegal. Qualquer ação que reduza v abaixo de 0 ou acima de uma velocidade máxima Vmax também é ilegal. O objetivo do agente é encontrar um plano que o estacione (estacionário) no quadrado de saída usando o mínimo possível de ações (passos de tempo). Por exemplo: se o agente mostrado estava inicialmente estacionário, ele poderia primeiro virar para o leste usando (E), então mover um quadrado para leste usando Rápido, então mais dois quadrados para leste usando Rápido novamente. O agente terá, naturalmente, que desacelerar para girar.



(a) Se o grid é MxN, qual o tamanho do espaço de estados? Justifique. Você deve assumir que todas as configurações são atingíveis a partir do estado inicial.
(b) Qual é o fator de ramificação máximo deste problema? Você pode assumir que as ações ilegais são simplesmente não retornadas pela função sucessor. Justifique sua resposta.
(c) A distância de Manhattan a partir da localização do agente para a saida do labirinto é admissível? Justifique sua resposta.
(d) Proponha e justifique uma heurística admissível não trivial para este problema.