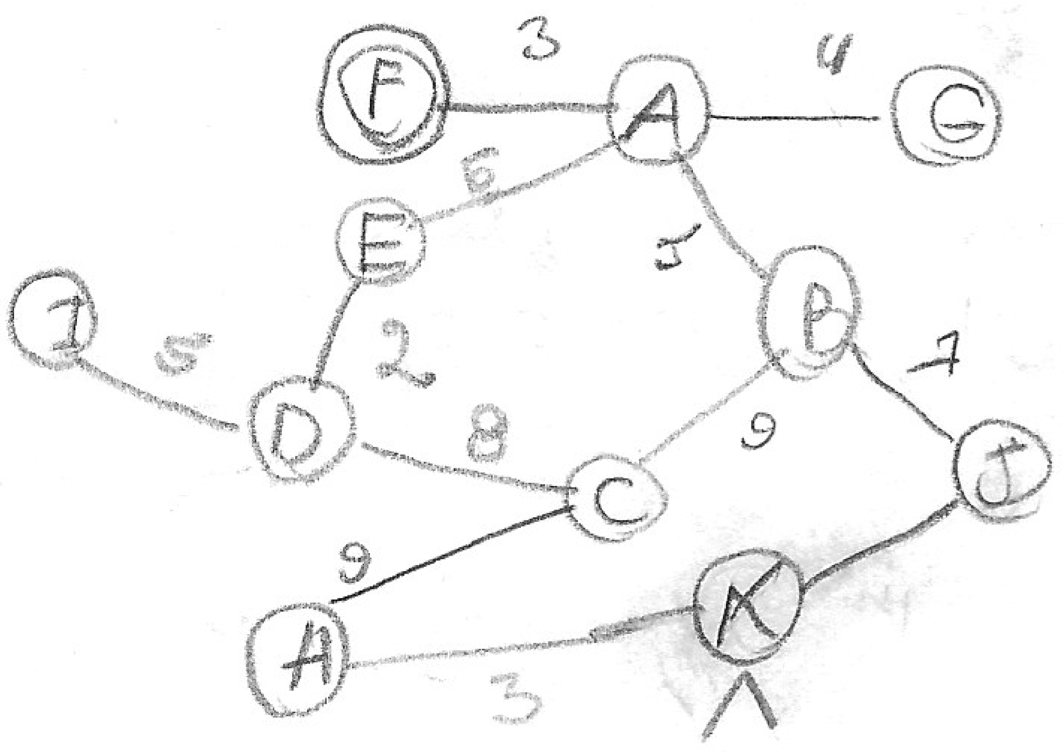
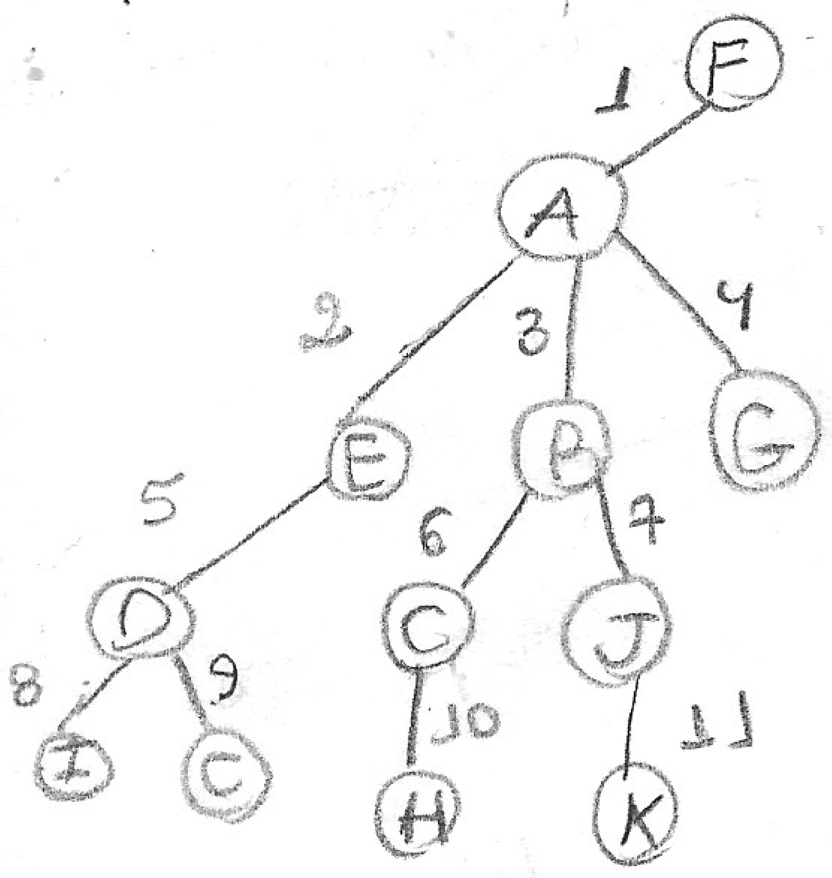
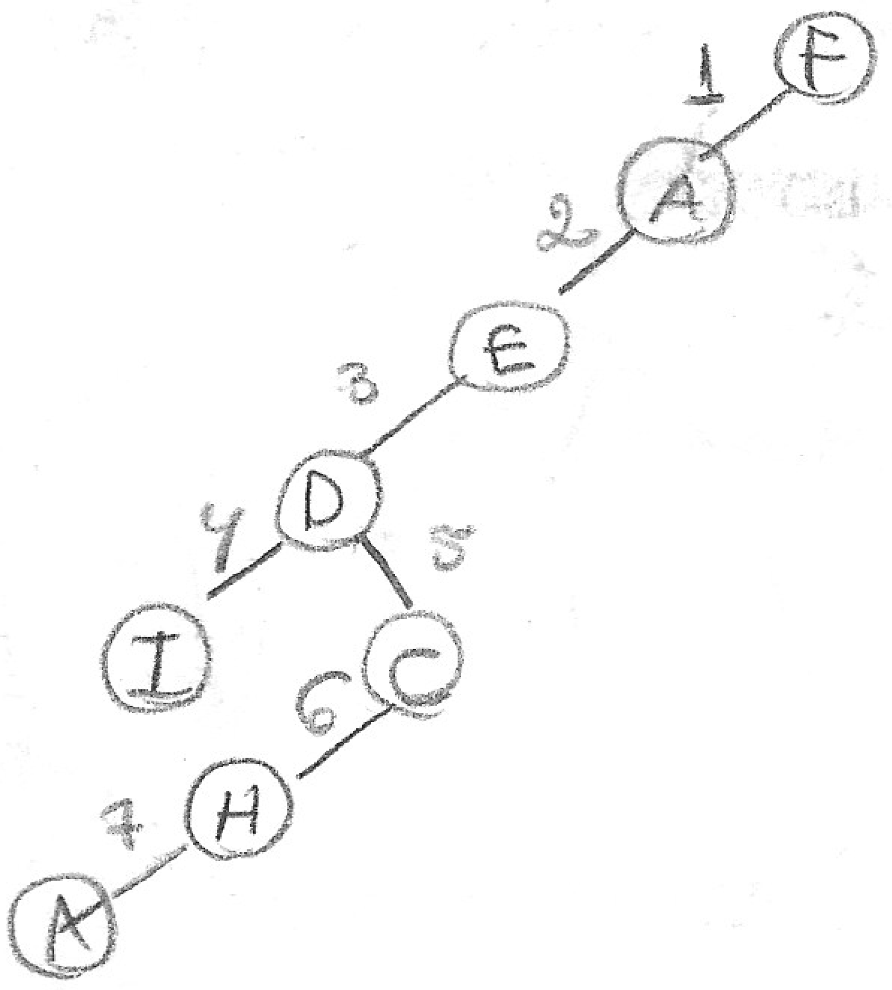
1

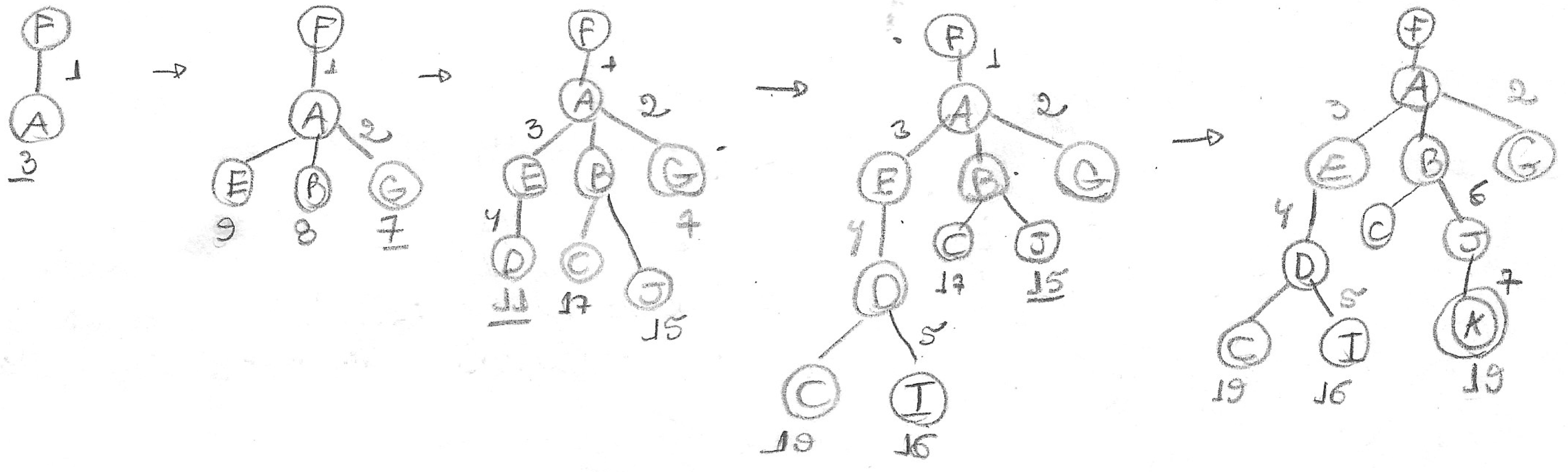


a) A seguir as árvores de cada algoritmo de busca. Os números são a ordem de criação. À esquerda, a busca por largura. À direita, profundidade

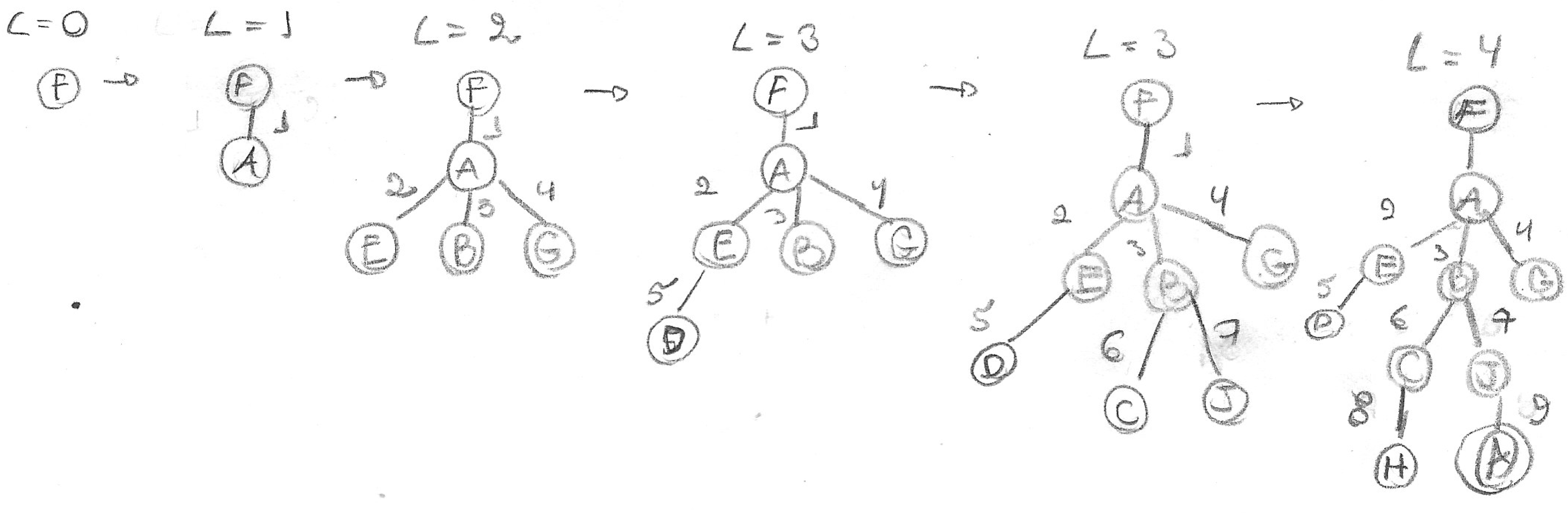




A busca em custo uniforme. Nesse caso, os valores na linha são a ordem de criação e os valores logo abaixo dos nós é o custo acumulado.

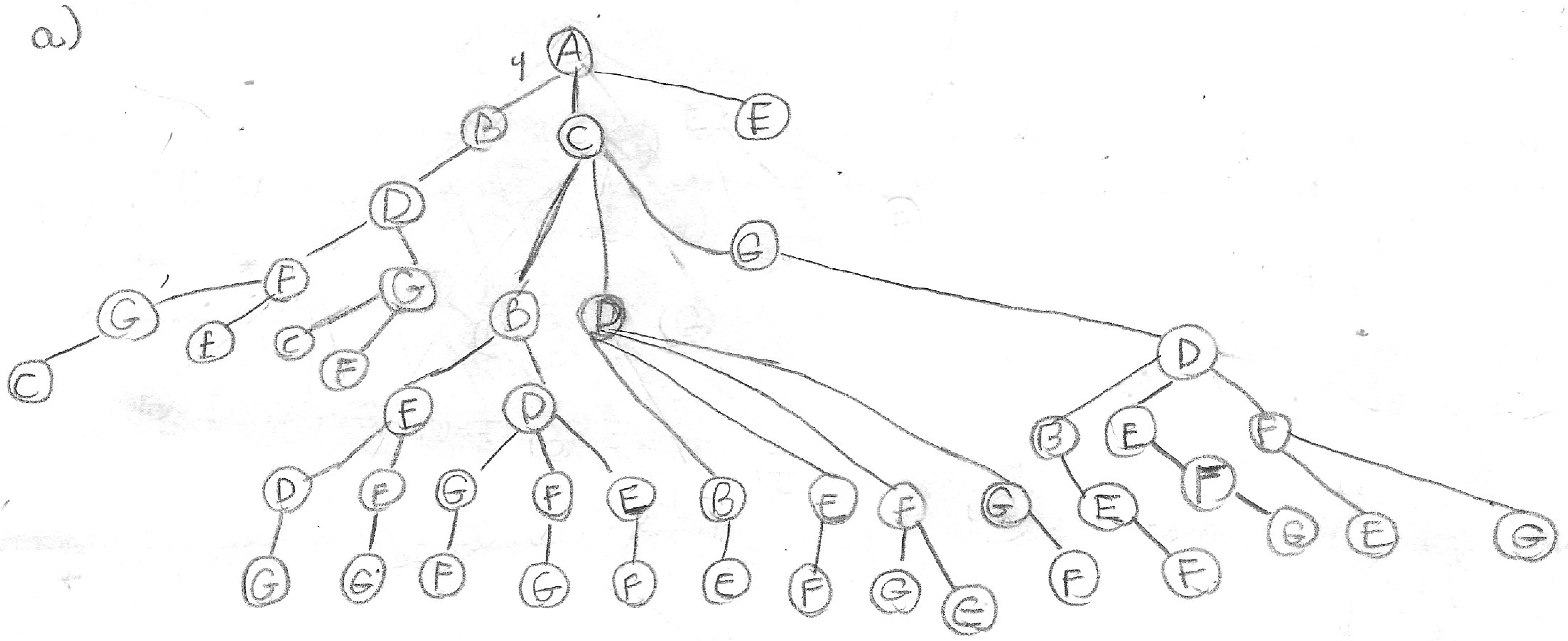


A busca com aprofundamento iterativo à seguir:



b) O melhor algoritmo em quantidade de nós avaliados foi o de busca uniforme. Mas nesse quesito ele empatou com a busca em profundidade. O algoritmo com o menor custo de caminho foi também o de custo uniforme, mas a busca por profundidade iterativa chegou no mesmo resultado: F, A, B, J, K.

② a)

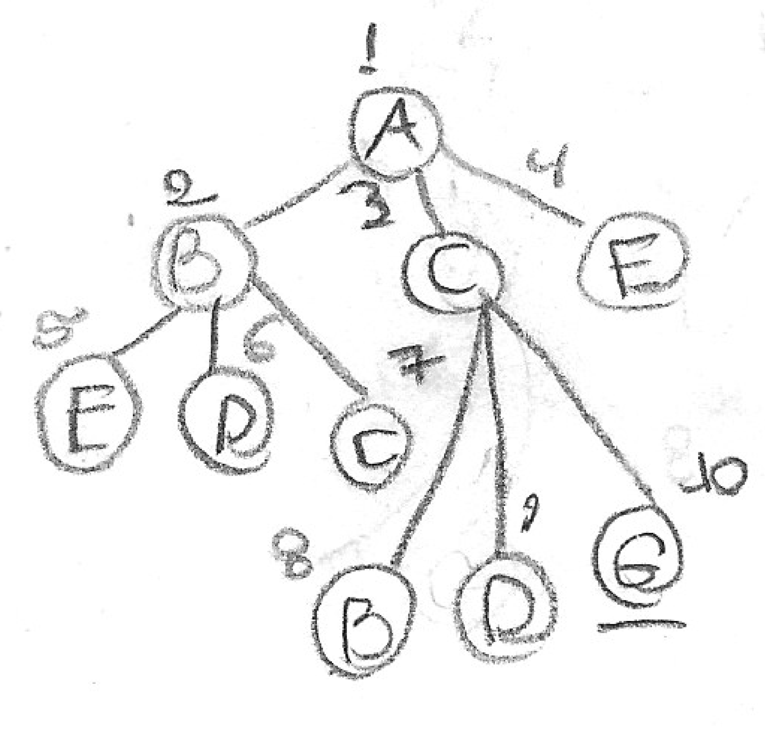


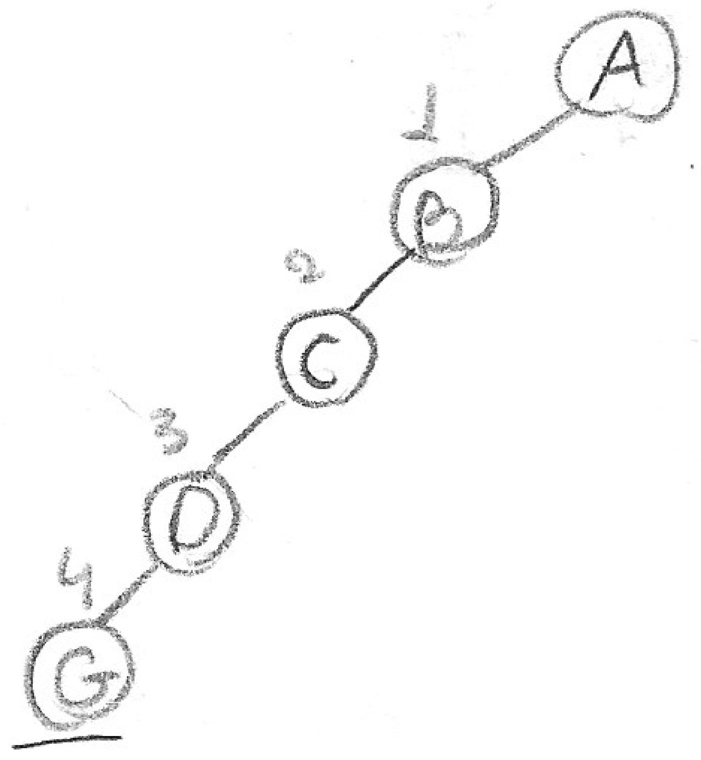
h(n) é o custo estimado do nó n até o nó solução. Ou seja, essa é a função heurística. Como o nó solução é G, temos:

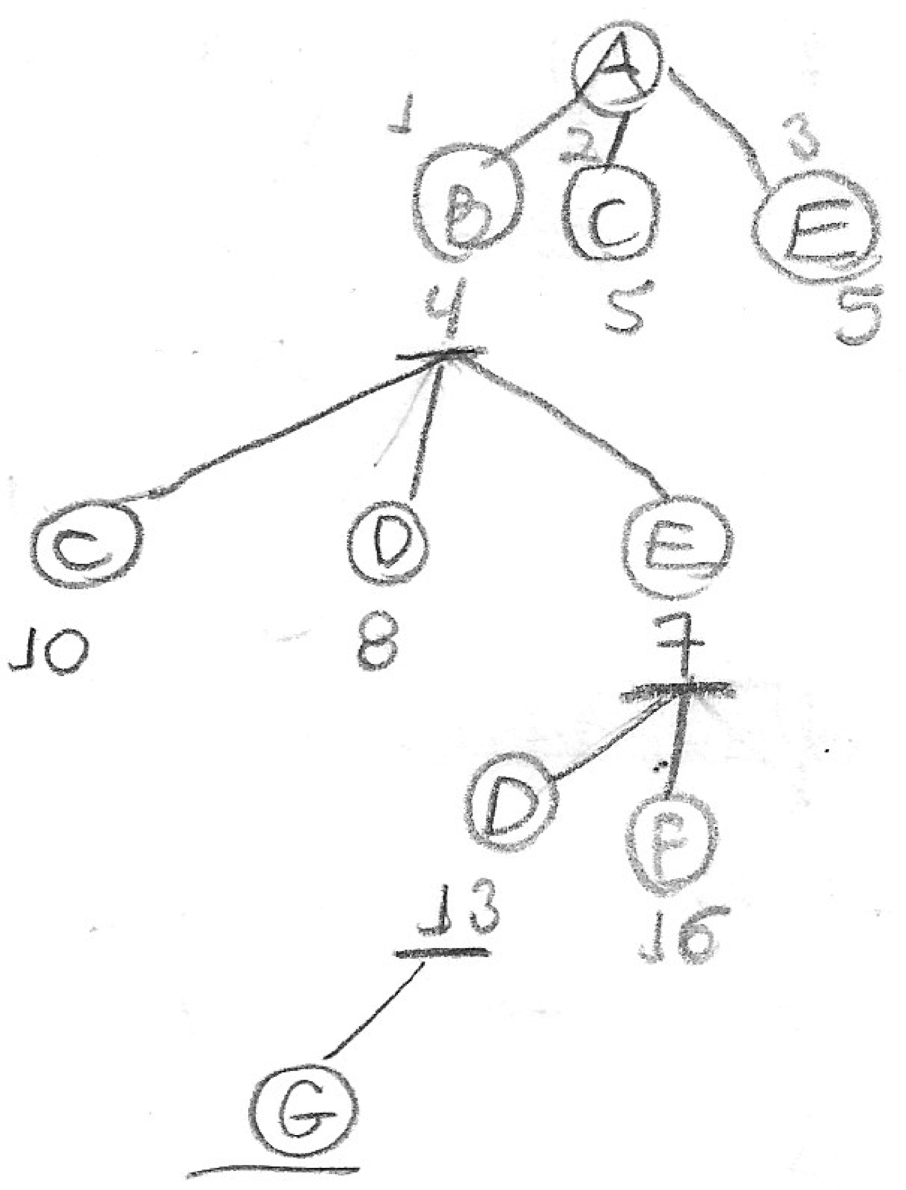
h(A) = 14, h(B) = 10, h(C) = 10, h(D) = 6, h(E) = 14, h(F) = 3

A solução ótima é A → B → D → G ou A → E → D → G, pois ambos possuem o mesmo valor.

b) Em largura: Em profundidade: em custo uniforme:

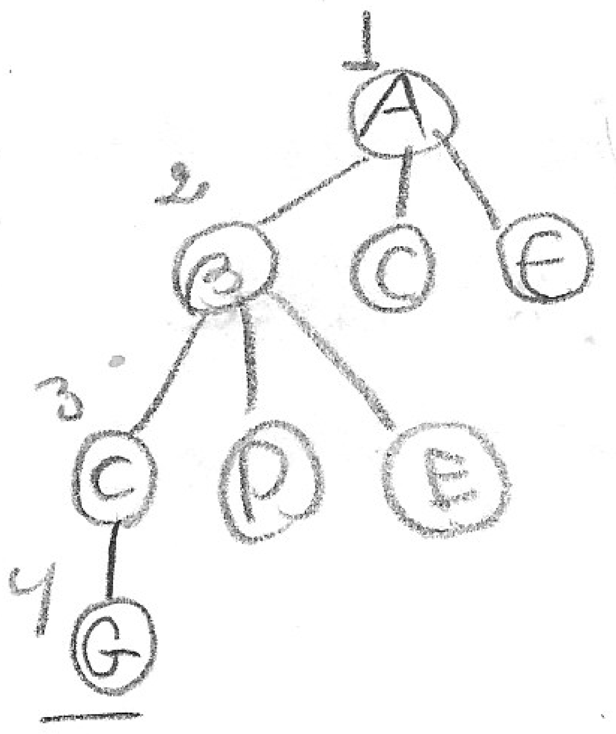






OBS: existe custo uniforme em profundidade? Porque foi basicamente isso que eu fiz.

Busca gulosa, usando como $h(n)$ a distância direta:



$h(A) = 3$

$h(B) = 2 = h(C) = h(E)$

$h(C) = 1$

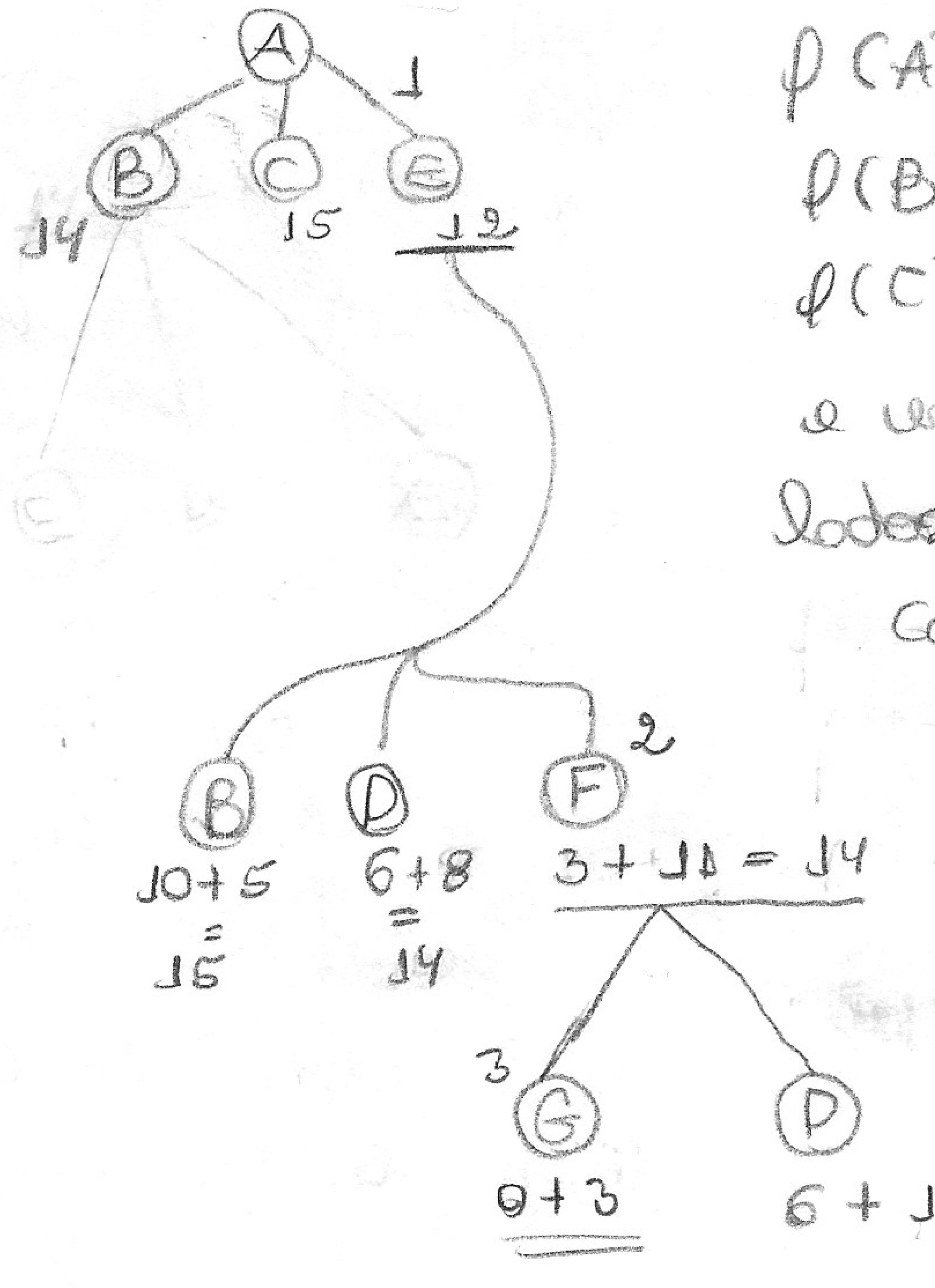
$h(D) = 1$

$h(E) = 2$

Busca A*

Uma busca com A* usa uma heurística admissível. Ou seja, $h(n) \leq h^*(n)$, onde $h^*(n)$ é o custo verdadeiro. É dito heurística admissível porque não superestima o valor de $h^*(n)$ que representa o verdadeiro custo

Disso, vou usar como heurística nesse caso o primeiro custo total do nó n até o nó solução (G) seguindo-se a ordem alfabética dos nós. Disso, somo $h(n)$ com o custo dos nós (acumulados



$f(A) = 15+0=15$ $f(E) = 12+2 = 12$

$f(B) = 10+4 = 14$

$f(C) = 10+5 = 15$

e em seguida recalcula com os custos acumulados.

Como houveram vários casos em que deu 14, (B, D, F) usei como critério para escolha a aleatoriedade.

c) Nenhuma delas encontrou a melhor solução, ou alguma das melhores soluções. Porém, não realizei todo o processo para uma busca completa nos algoritmos de busca em largura, profundidade, custo uniforme. Ou seja, esses algoritmos que fazem a busca completa conseguem garantir a solução ótima. A busca gulosa não é completa então não garante a solução ótima. A busca em A* pode ser completa. O número de nós explorados está enumerado nos grafos.

③ Para facilitar, vou chamar o recipiente 1 de A e o recipiente 2 de B.

a) Estado inicial: (A, 0), (B, 0). Significa que cada um dos recipientes tem 0 litros. O estado final será (A, 2), (B, X) onde X é um valor qualquer porque não estamos interessados na sua quantidade ao final.

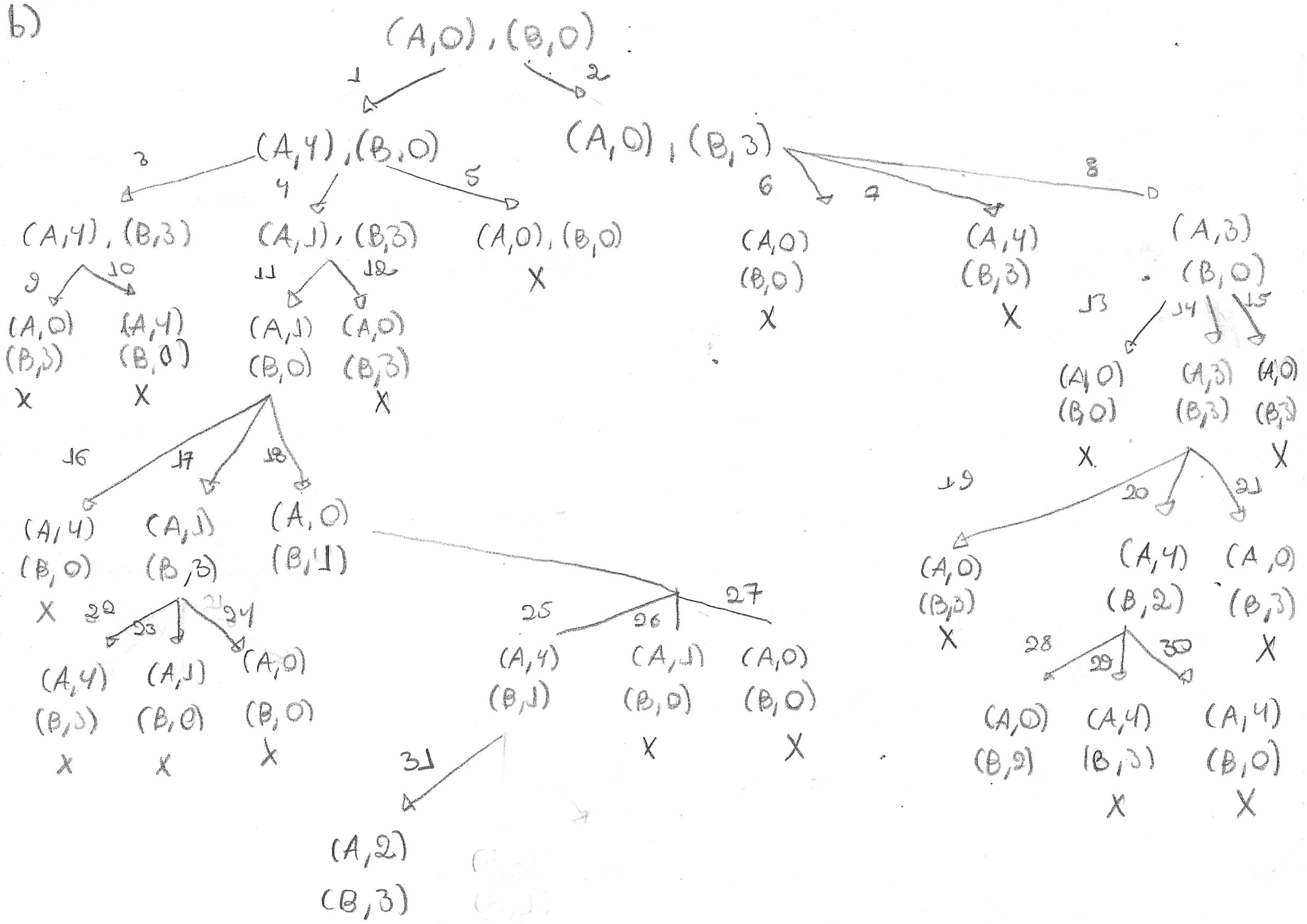
As ações podem ser:

encher o recipiente A: (A,0), (B,0) → (A,4), (B,3)

esvaziar o recipiente: (A,0), (B,3) → (A,0), (B,0)

transferir para o recipiente: (A,0), (B,3) → (A,3), (B,0)

b)



Com busca em largura, encontramos a solução em 31 nós explorados: (A,0),(B,0) → (A,4),(B,0) → (A,1),(B,3) → (A,1),(B,0) → (A,0),(B,1) → (A,4),(B,1) → (A,2),(B,3) □

Questão (4) :

a) O estado inicial (q_0) é dado pela tupla (1,1). O espaço de estados são tuplas (m, n), sendo definidos de (1,1), (1,2), (1,3) ... (4,5), (5,5). Ou seja, cada tupla marca uma posição na matriz 5x5 do labirinto. O estado final (q_F) é dado por (1,5)

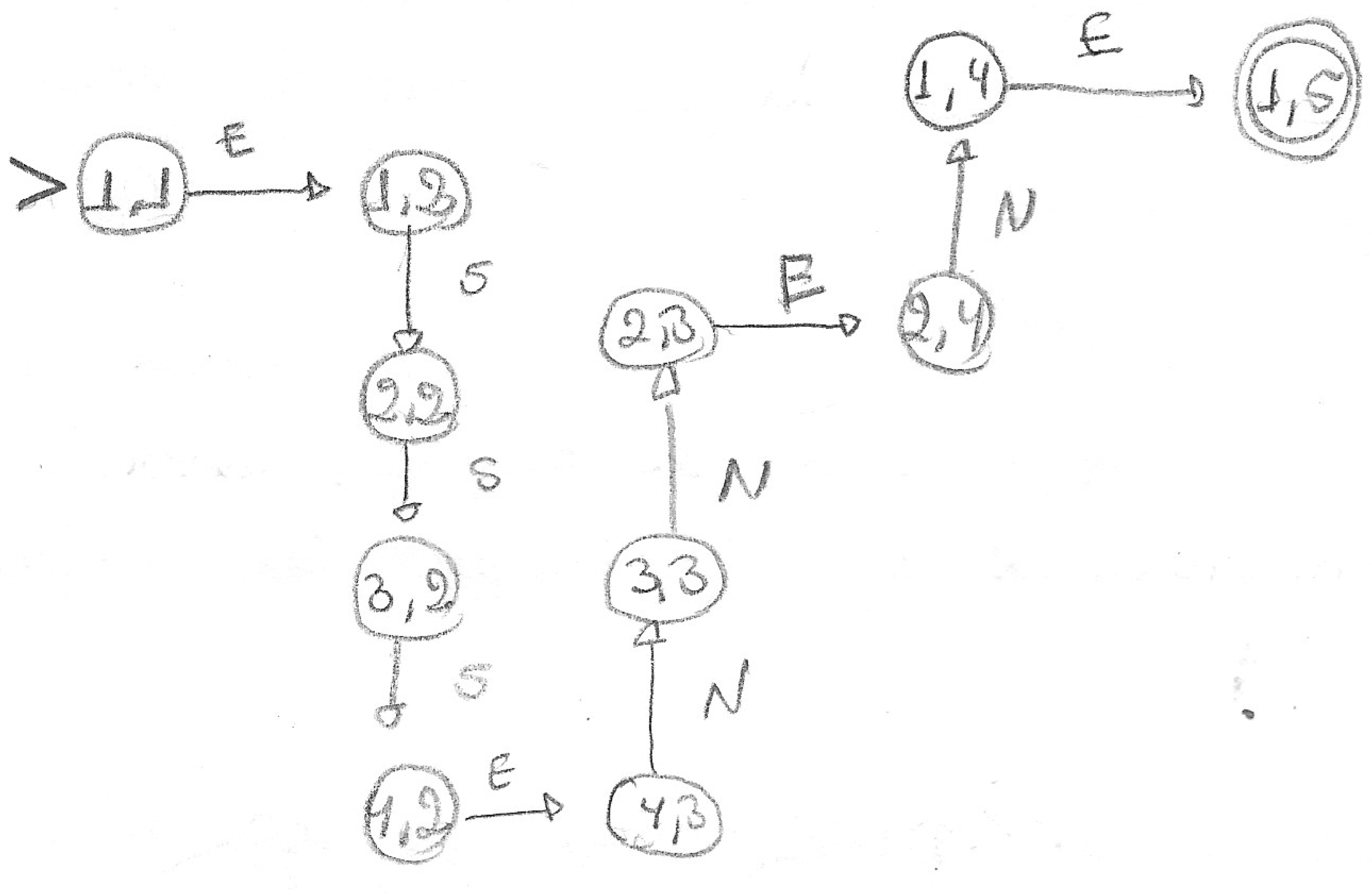
As funções de transição são as seguintes

$f((m,n), N) = (m, n+1)$ | $f((m,n), W) = (m+1, n)$

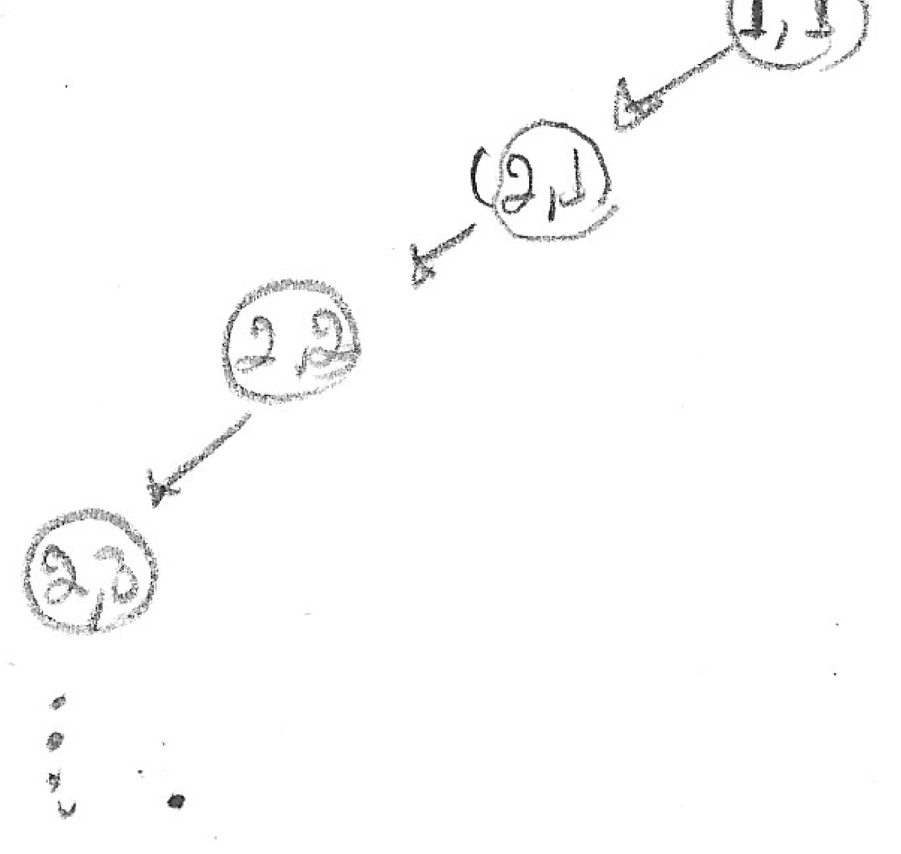
$f((m,n), S) = (m, n-1)$ | $f((m,n), E) = (m-1, n)$

Lembrando que quando há barreiras entre as posições, as transições são inválidas, assim como quando o estado seguinte está fora dos limites do labirinto

b)



c) Uma busca em profundidade, se a escolha dos nós for organizada em um grafo, escolherá o nó mais à esquerda no grafo até que se esgotem as opções. Assim como está desenhado à esquerda.



A busca em profundidade é ótima porque é capaz de encontrar o caminho de menor custo. Mas ela também pode finalizar abruptamente assim que encontra um caminho até o fim, se for necessário.

d) A busca em largura explora os opções uma de cada vez, assim que os encontra nos nós vizinhos. Essa busca também é ótima porque se ela for capaz de explorar todos os nós, encontrará o melhor caminho.

Questão 5

a) Vou assumir que tem-se apenas 2 processadores, como foi visto em sala de aula.
Um estado é dado por dois processadores (P1 e P2) e suas tarefas alocadas. Por exemplo:

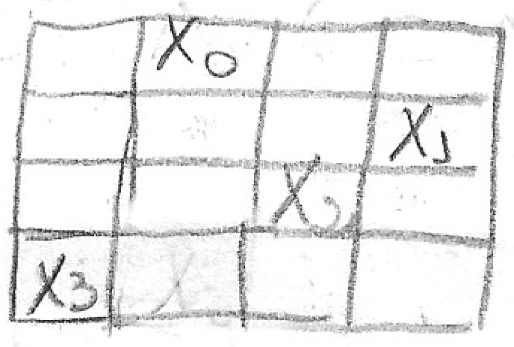
P1 = {1}
P2 = {2}
} Também podendo ser uma tupla ([1], [2])

O estado inicial é aquele em que os dois processadores estão vazios
O estado final é aquele em que todas as tarefas foram alocadas respeitando a ordem de dependências.

b) Uma função de custo nesse problema, ao ler um estado, devolve o tem total para a finalização de todas as tarefas, levando em consideração o tempo de comunicação e a paralelização das tarefas quando for possível.

c) Uma heurística admissível seria somar o tempo das tarefas e dividir pela quantidade de processadores. Esse método daria um valor bem otimista, já que assume uma paralelização perfeita entre o tempo de execução dos processadores. Ou seja, nunca superestima o valor verdadeiro e por isso é admissível.

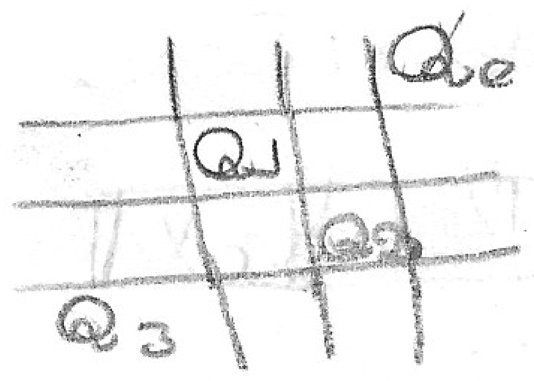
⑥



→ estado inicial de exemplo

Só é possível trocar a posição da coluna das rainhas.

questão 1) Para transicionar entre os estados, podemos mudar a posição das colunas de duas rainhas.



No exemplo à esquerda, podemos mudar as seguintes para em uma transição: (Q_0, Q_1), (Q_0, Q_2), (Q_0, Q_3), (Q_1, Q_2), (Q_1, Q_3), (Q_2, Q_3). Portanto, existem 6 vizinhos para um estado.

questão 2)

Seja Q_i uma representação dos estados em uma tupla (X_0, X_1, X_2, X_3). Assim sendo, o estado inicial é $Q_0 = (3,1,2,0)$. Sendo a função de custo o número de rainhas em conflito, temos $f(Q_0) = 2$. Queremos minimizar esse valor.

Dentre os sucessores, escolheremos o seu melhor. Vamos ver as possibilidades de trocar a rainha X_0:

$(\underline{3},1,2,0) \to (1,3,2,0)$ *, $(2,1,3,0)$, $(0,1,2,3)$

$(1,3,\underline{2},0) \to (1,3,0,2)$, $(2,3,1,0)$, $(1,2,3,0)$

$f((1,3,2,0)) = 1$ *
$f((2,1,3,0)) = 1$
$f((0,1,2,3)) = 3$

$f((1,3,0,2)) = 0$ □
$f((2,3,1,0)) = 2$
$f((1,2,3,0)) = 2$

Com duas etapas de escolher sucessores chegamos à solução $(1,3,0,2)$ que é ótima. Também foi feita exatamente a troca da menor coluna 1 com 3 no lugar de 2 com 3 em caso de empate de custo.

questão 3)

O algoritmo de subida de encosta nesse caso consiste em encontrar o estado com o menor número de conflitos. Portanto, temos um problema de minimização.

Temos então o estado (3,2,0,1).

O custo desse estado é 2. Esse estado é um ótimo local se nenhuma das próximas transições oferecer melhora (valor menor).

$f((2,3,0,1)) = 4$,

$f((0,2,3,1)) = 1$

vemos já que o estado (0,2,3,1) é uma transição possível e tem um custo menor. Portanto (3,2,0,1) não é um máximo local. Além do mais, mesmo (0,2,3,1) sendo um estado melhor ele não é um máximo global.

questão 4) O custo do estado (0,0,0,0) é 3. Se considerarmos que a obtenção dos vizinhos é feita trocando-se as posições das colunas de duas rainhas entre si, como todas as rainhas estão na mesma coluna, todos os estados vizinhos serão (0,0,0,0). Assim sendo, não tem vizinhos com custo menor e portanto (0,0,0,0) é um máximo local. Mas claro, bastando trocar a regra para obtenção de vizinhos e isso não mais será verdade.

⑦ a) Podemos representar esse problema como um vetor de 8 posições: $\underset{s}{\_\_}\ \underset{e}{\_\_}\ \underset{n}{\_\_}\ \underset{d}{\_\_}\ \underset{m}{\_\_}\ \underset{o}{\_\_}\ \underset{r}{\_\_}\ \underset{y}{\_\_}$, cada uma representando uma letra.

b) O estado inicial pode ser dado por uma string de 8 números como 12345678. O estado meta é aquele em que a soma dos números posicionados nas letras "send" e "more" resultam em um número construído pelas letras "money"

c) Usando como exemplo a string 12345678, vou assumir que cada vizinho pode ser obtido alterando-se um número de alguma das posições. Como são 8 posições e em cada uma é possível escolher entre 10 números, um estado possui 80 vizinhos.

d) A função de custo é a diferença entre a palavra "money" e o cálculo do estado atual em "send" + "more". Se essa diferença for Ø, esse é o estado meta. (e vale ressaltar que a diferença tem que ser especificamente uma diferença absoluta)

⑦ 2)

Array de estado inicial: $\frac{2}{c}\ \ \frac{3}{o}\ \ \frac{1}{a}\ \ \frac{7}{l}\ \ \frac{4}{a}\ \ \frac{5}{i}$

aplicando a função custo: $f(q_1) = |\underbrace{31454}_{\text{coisa}} - (\underbrace{2321}_{\text{coca}} + \underbrace{2371}_{\text{cola}})|$

$f(q_1) = 26762$

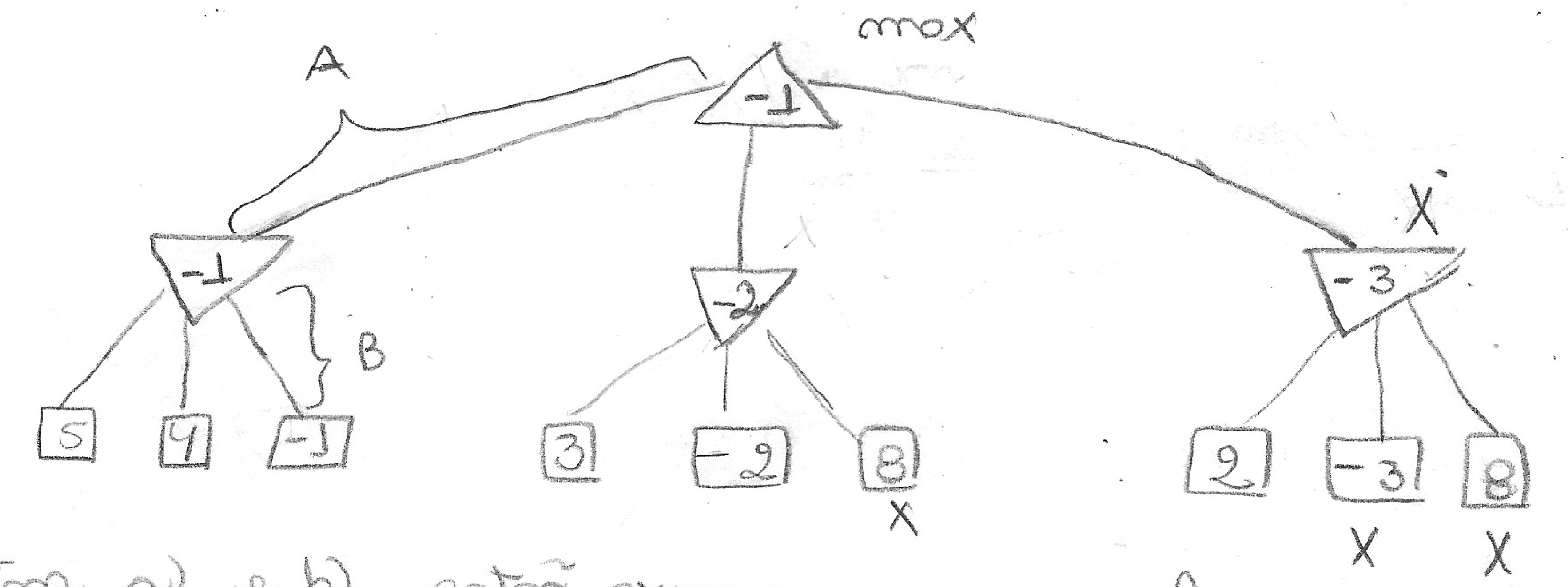
Queremos um próximo estado com um custo menor.

Possível vizinho: 931745

$f(q_2) = |31454 - (9391 + 9371)| = 12692$

e isso se segue até que a diferença seja 0

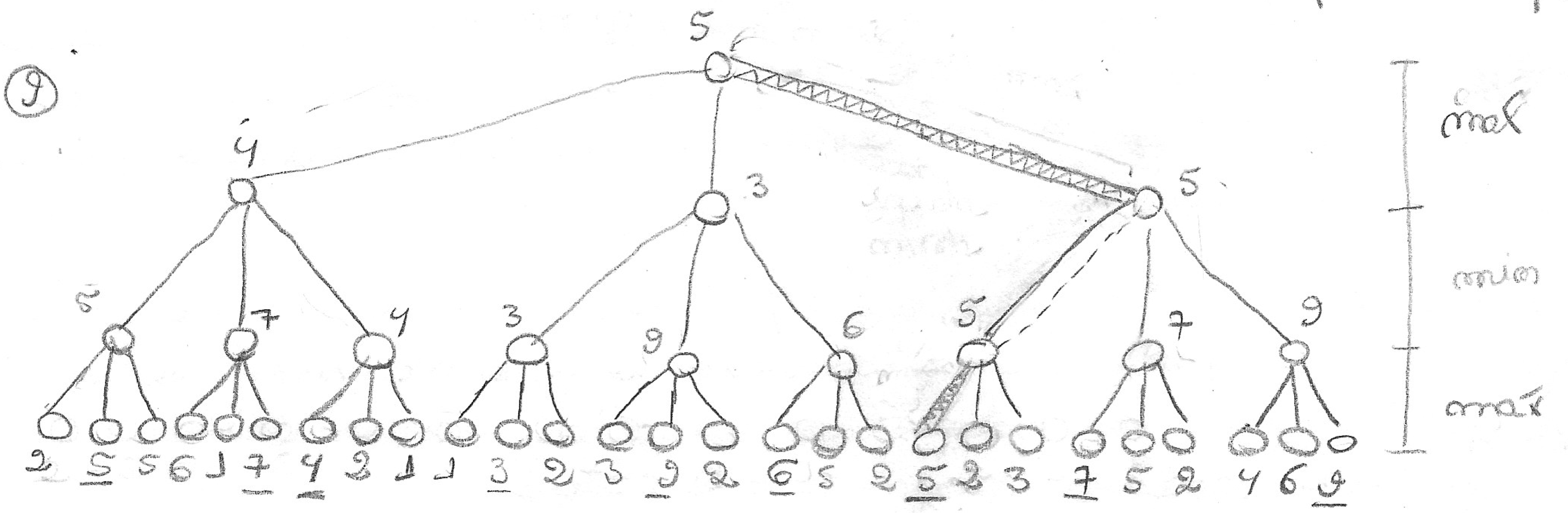
⑧



letras a) e b) estão expressas no grafo.

pela letra c), podei os nós segundo a poda alfa.

⑨



legenda:

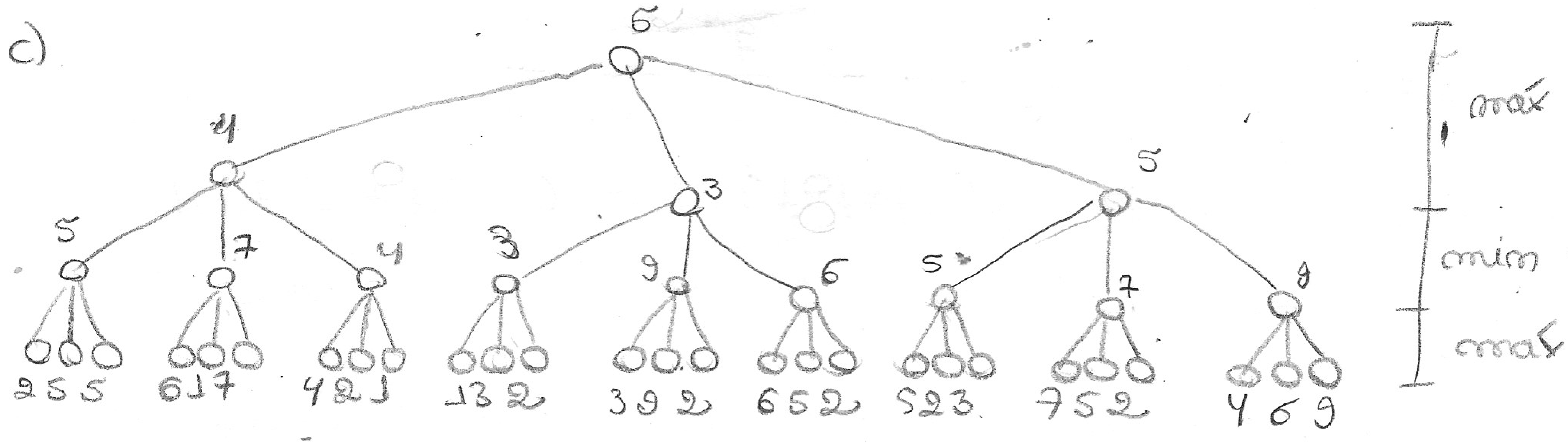
// = jogada do minimizador

= jogada do maximizador.

b) O maximizador receberá o valor 3 após 2 lances

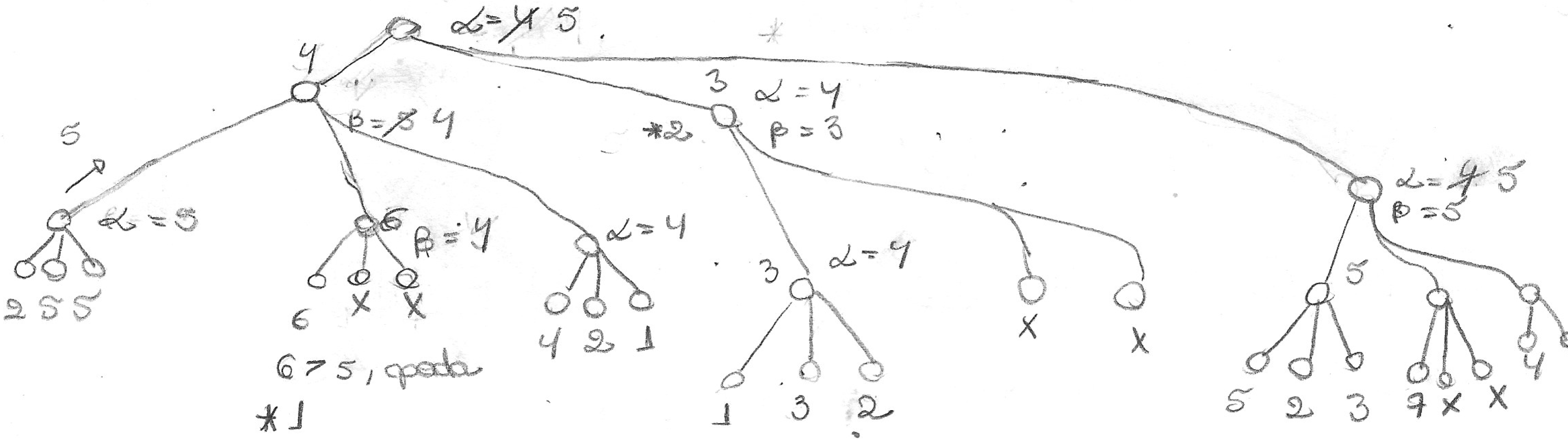
⑨ Percorrendo a árvore:

c)



A exploração dos nós na árvore foi feita da esquerda para a direita.

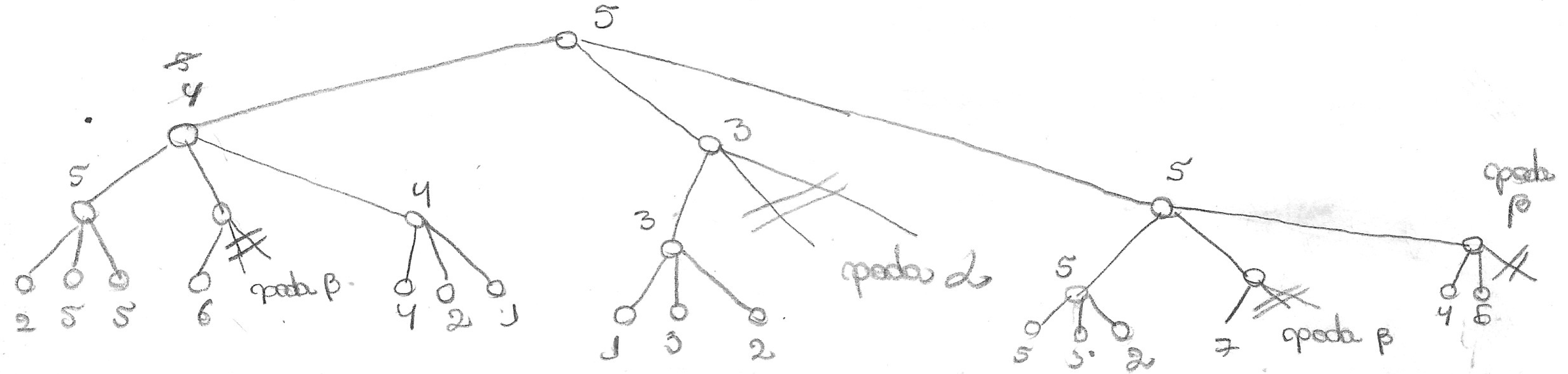
Na poda α-β os diferentes níveis máx/min em profundidade α é armazenado para cada nó máx enquanto β é armazenado para cada nó min



*1: Aqui foi feita uma poda β. Como já no primeiro nó à esquerda foi encontrado um valor maior que o β atual, podemos podar. A ideia é que mesmo se houver um nó com valor maior nesse maximizador mais profundo, o nó acima é minimizador e portanto não está interessado nele.

*2: Aqui, como achamos um 3 e o $\alpha = 4$, sabemos que o maximizador do topo não vai escolher esse caminho de jeito nenhum. Isso porque o minimizador com valor 3 só vai achar valores menores que isso, e portanto podemos podar seus 2 netos.

A árvore final com as podas fica assim:



(10)

a) não visitei o Rio de Janeiro então não gosto de viajar

b) se gosto de e não visitei o Rio de Janeiro, então não gosto de viajar.

c) visitei o Rio de Janeiro e não gosto de viajar.

d) não visitei o Rio de Janeiro e nem gosto de viajar.

Agora, descrevendo as sentenças:

a) $P = 1 > 2$, Q = qualquer coisa é possível

$P \rightarrow Q$.

b) P = elefantes podem subir em árvores Q - 3 é um número racional

$P \rightarrow Q$

c) P = proibido fumar cigarro, Q = proibido fumar charuto

$P \vee Q$

d) $P = x > 0$, $Q = x > 1$

$\neg (P \leftrightarrow Q)$

(11)

a) Eis a nossa BC:

Quero esclarecer antes que estou usando a notação $P_{c,l}$, onde c é a coluna e l a linha. Assim, temos:

$B_{1,4} \leftrightarrow (P_{2,4} \vee P_{1,3})$: R_1

$B_{2,3} \leftrightarrow (P_{2,4} \vee P_{3,3} \vee P_{2,2} \vee P_{1,3})$ R_2

$S_{2,2} \leftrightarrow (W_{1,2} \vee W_{2,1} \vee W_{3,2} \vee W_{2,3})$: R_3

$B_{3,1} \leftrightarrow (P_{2,1} \vee P_{3,2} \vee P_{4,1})$: R_4

$S_{3,1} \leftrightarrow (W_{2,1} \vee W_{3,2} \vee W_{4,1})$: R_5

$\neg P_{1,3}$, $\neg W_{1,3}$, $\neg S_{1,3}$ | $\neg P_{1,1}$, $\neg W_{1,1}$, $\neg S_{1,1}$

$\neg P_{1,2}$, $\neg W_{1,2}$, $\neg S_{1,2}$ | $\neg P_{2,1}$, $\neg W_{2,1}$, $\neg S_{2,1}$.

b.1)

Marquei as regras na questão anterior e vou usá-las para inferir a posição do Wumpus.

$$S_{2,2} \leftrightarrow (W_{3,2} \vee W_{2,1} \vee W_{1,2} \vee W_{2,3})$$

Removendo $\wedge$ / $\leftrightarrow$:

$$W_{3,2} \vee W_{2,1} \vee W_{1,2} \vee W_{2,3} \rightarrow S_{2,2}$$

Aplicando $\neg W_{1,2}$, $\neg W_{2,1}$, $\neg_{2,3}$:

$$W_{3,2} \rightarrow S_{2,2}$$

Aplicando modus ponens em $S_{2,2}$:

$W_{3,2}$ $\square$

b 2) Ele sabe que existe um poço com certeza em $P_{4,2}$ e em $P_{1,4}$ (A notação aqui é $P_{linha,\, coluna}$)

$$\begin{array}{l} B_{1,3} \rightarrow P_{2,3} \vee P_{1,4} \vee P_{1,2} \\ \qquad\qquad\qquad\qquad \neg P_{1,2} \\ \hline \end{array}$$

$$\left.\begin{array}{l} B_{1,3} \rightarrow P_{2,3} \vee P_{4,4} \\ S_{1,3} \wedge S_{2,2} \rightarrow W_{2,3} \\ W_{2,3} \rightarrow \neg P_{2,3} \end{array}\right\} \qquad \begin{array}{c} B_{1,3} \rightarrow P_{2,3} \vee P_{1,4} \\ \neg P_{2,3} \\ \hline B_{1,3} \rightarrow P_{1,4} \quad \square \end{array}$$

notação $W_{coluna,\, linha}$

c) Atirar uma flecha em direção à $W_{3,2}$. É a localização exata, e matando o Wumpus o agente ganha o jogo.