Chapter3

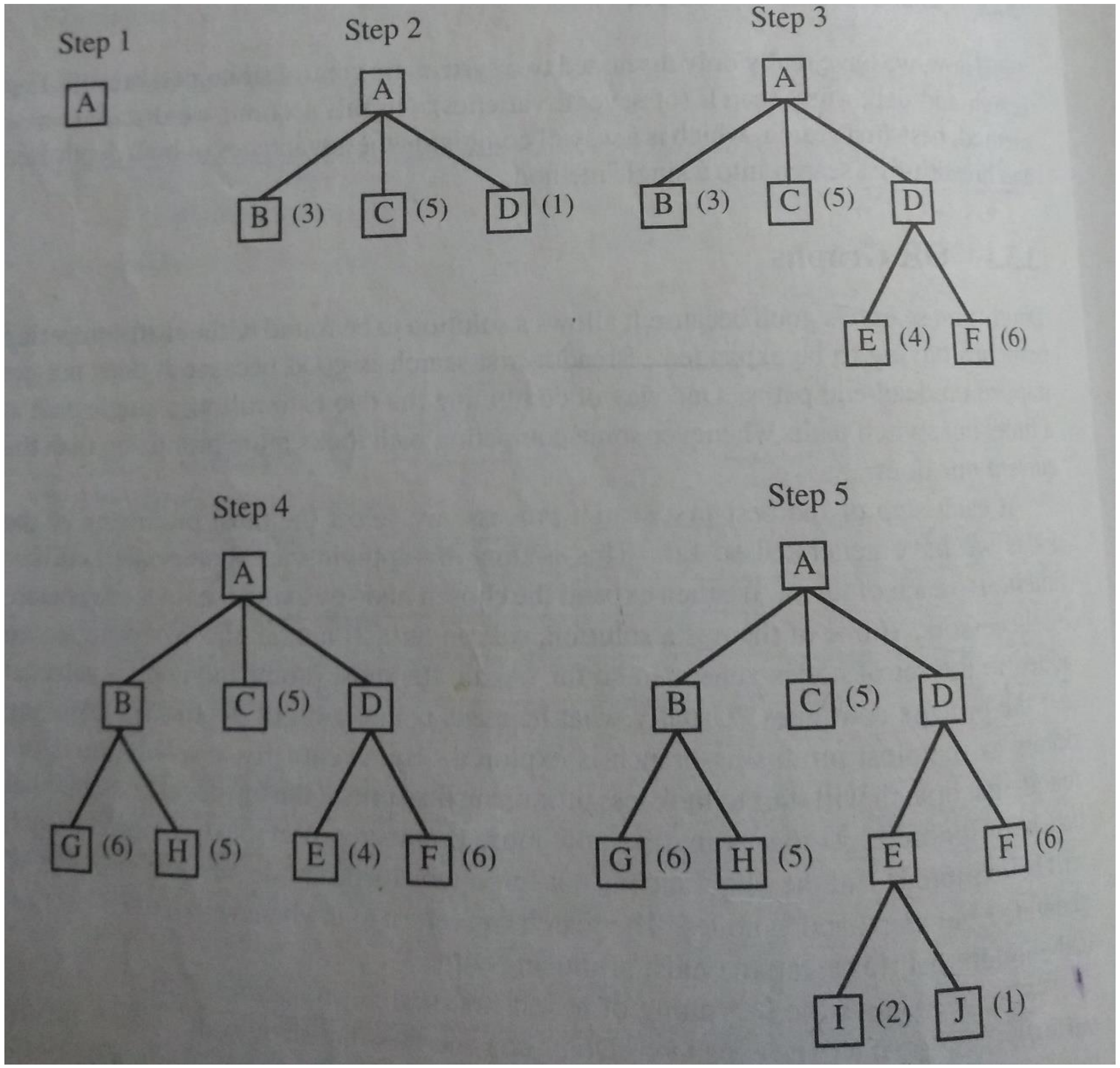
Heuristic Search Techniques

(Cont….)

Best First Search

- Way of combining the advantages of both breadth first search and depth first search
- DFS: Good, as it allows a solution to be found without all competing branches having to be expanded
- BFS: Good, as it does not get trapped on dead end paths
- Best First Search: Follows a single path at a time, but switch path when some competing path looks better

Best First Search



Best-First Search V/S Steepest ascent hill climbing

1. In hill climbing, one move is selected and all the others are rejected, never to be reconsidered

   In best-first search, one move is selected but the others are around so that they can be revisited later if required

2. If successor is having lower value than current state than hill climbing halts while

   best-first search selects the best possible state even though it is worst than current state

Graph Search

- To avoid duplication of nodes
- The information stored in each node structure:
  - Description of problem state it represents
  - Desirability: how promising a state is
  - A parent link: to recover solution path
  - List of successor: to propagate the improvements
- Algorithm operate by searching a directed graph
- Such a graph is known as ‘OR’ graph as each of its branches represents an alternative problem solving path

Lists of Nodes for Graph Search

- OPEN
  - Nodes that have been generated and have had the heuristic function applied to them but which have not yet been examined [generated but not explored]
  - Maintained in a priority queue
  - High priority→ high promising value
- Closed
  - Nodes that have already been examined
  - Kept in memory to avoid duplication of nodes

- **Heuristic function for evaluation:**

$$f' = g + h'$$

g : Measure of the cost of getting from initial state to the current node [ACTUAL]

h' : Estimation of additional cost of getting from the current node to a goal state

f' : Estimation of total cost of getting from the initial state to a goal state

Best-First Search

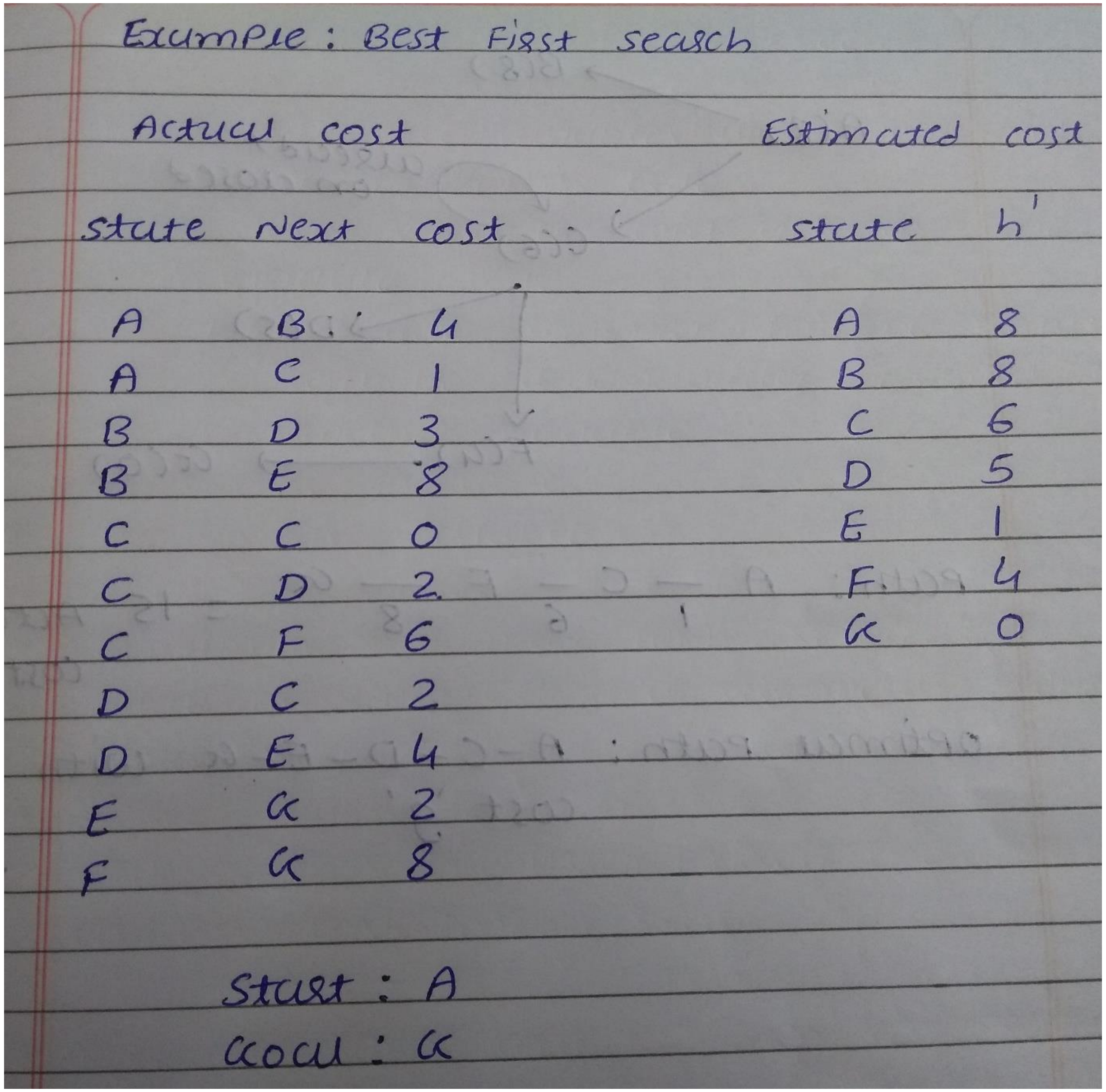
- **Algorithm:**

1. **Start with OPEN containing just the initial state**
2. **Until a goal is found or there are no nodes left on OPEN do:**
   a) **Pick the best node on OPEN**
   b) **Generate its successors**
   c) **For each successor do:**
      i. **If it has not been generated before, evaluate it, add it to OPEN and record its parent**
      ii. **If it has been generated before, change the parent if this new path is better than previous one. In that case, update the cost of getting to this node and to any successors that this node may already have**

Best-First Search

- **Note :**
  - **It selects vertex closest to the goal, does not consider distance from starting point**
  - **It considers only h’**
  - **Greedy in nature**

Best-First Search Example1



Example : Best First search

Actual cost

state	Next	cost
A	B	4
A	C	1
B	D	3
B	E	8
C	C	0
C	D	2
C	F	6
D	C	2
D	E	4
E	G	2
F	G	8

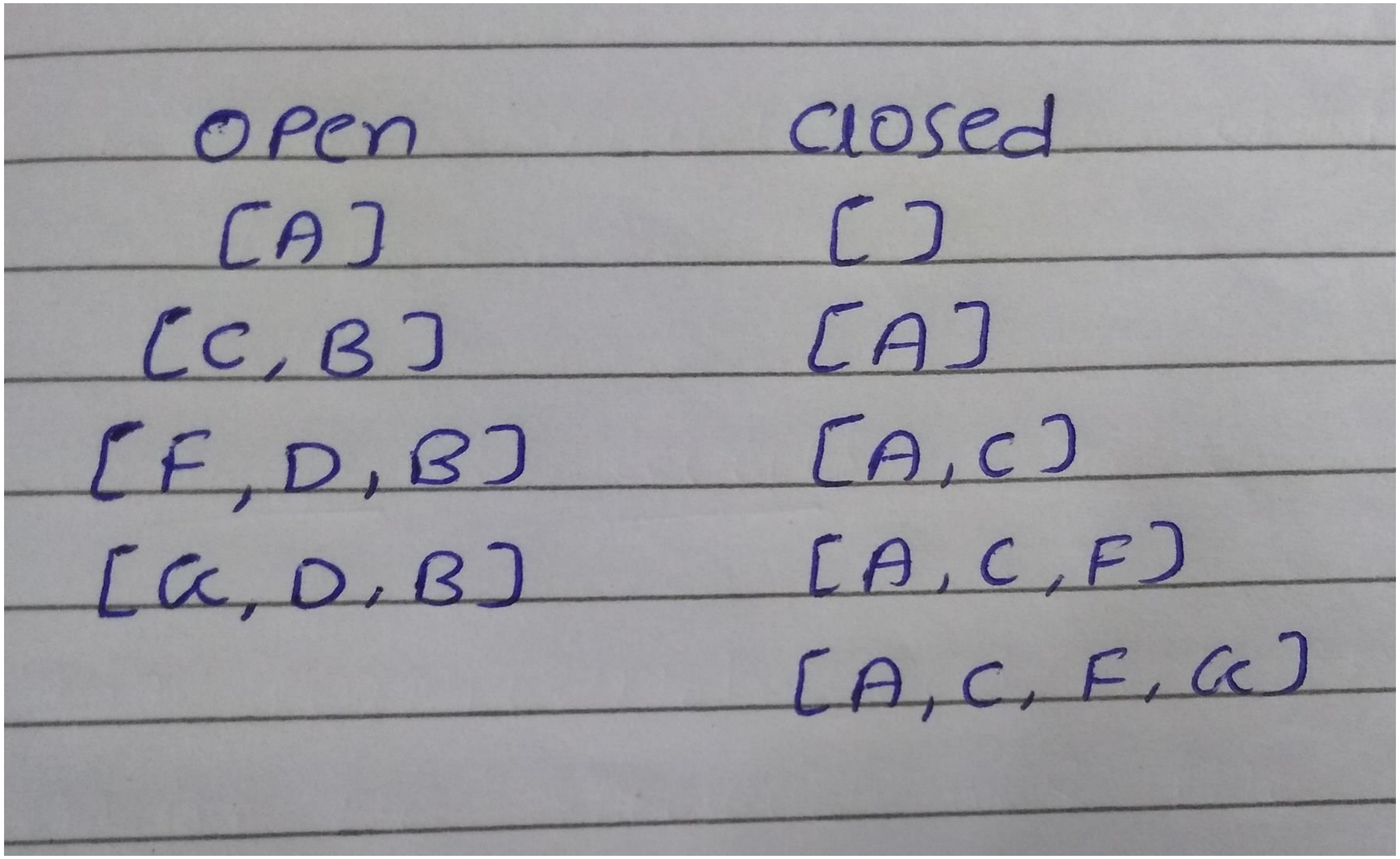
Estimated cost

state	h'
A	8
B	8
C	6
D	5
E	1
F	4
G	0

Start : A

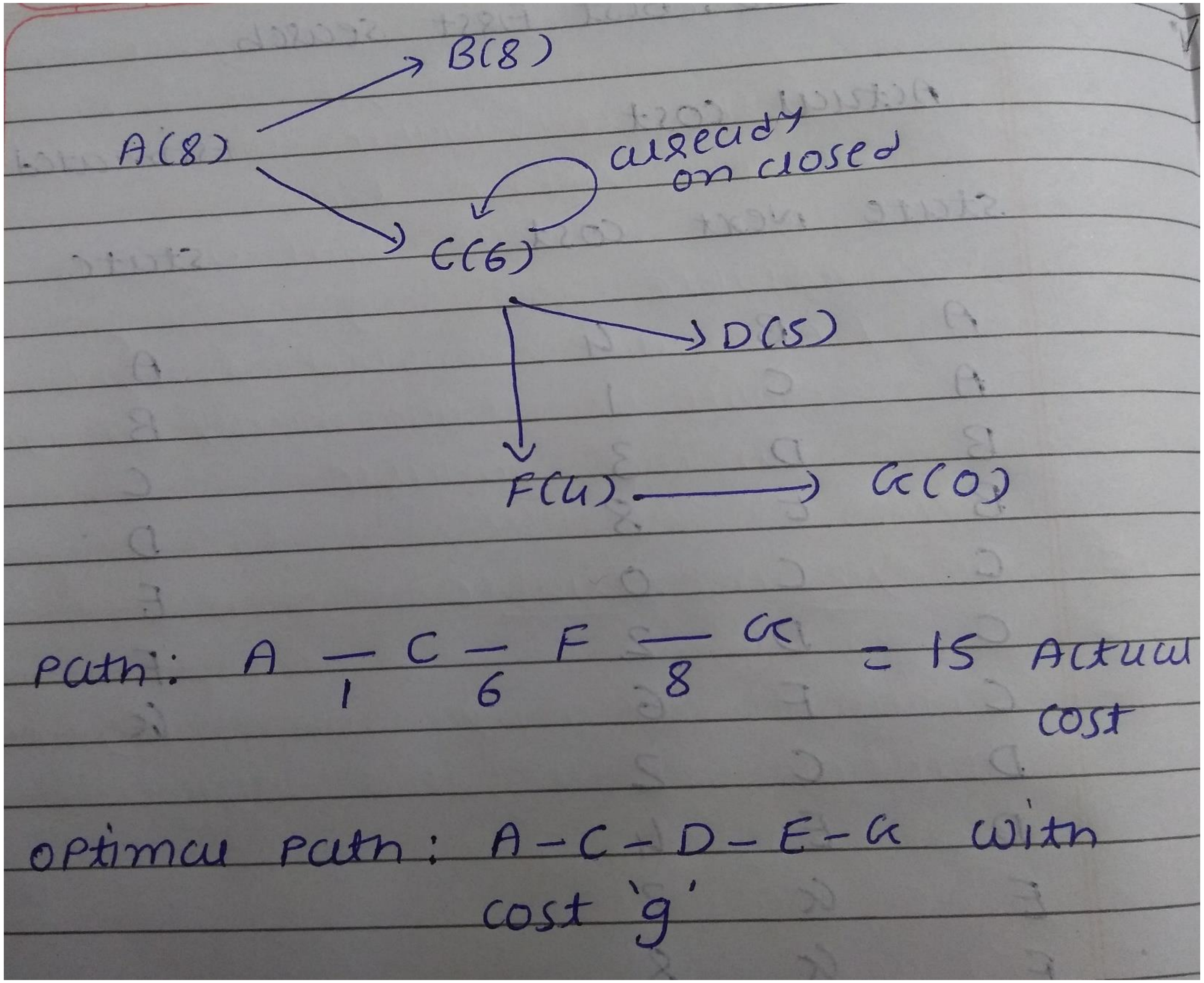
Goal : G

Best-First Search Example1



Open	Closed
[A]	[ ]
[C, B]	[A]
[F, D, B]	[A, C]
[G, D, B]	[A, C, F]
	[A, C, F, G]

Best-First Search Example1

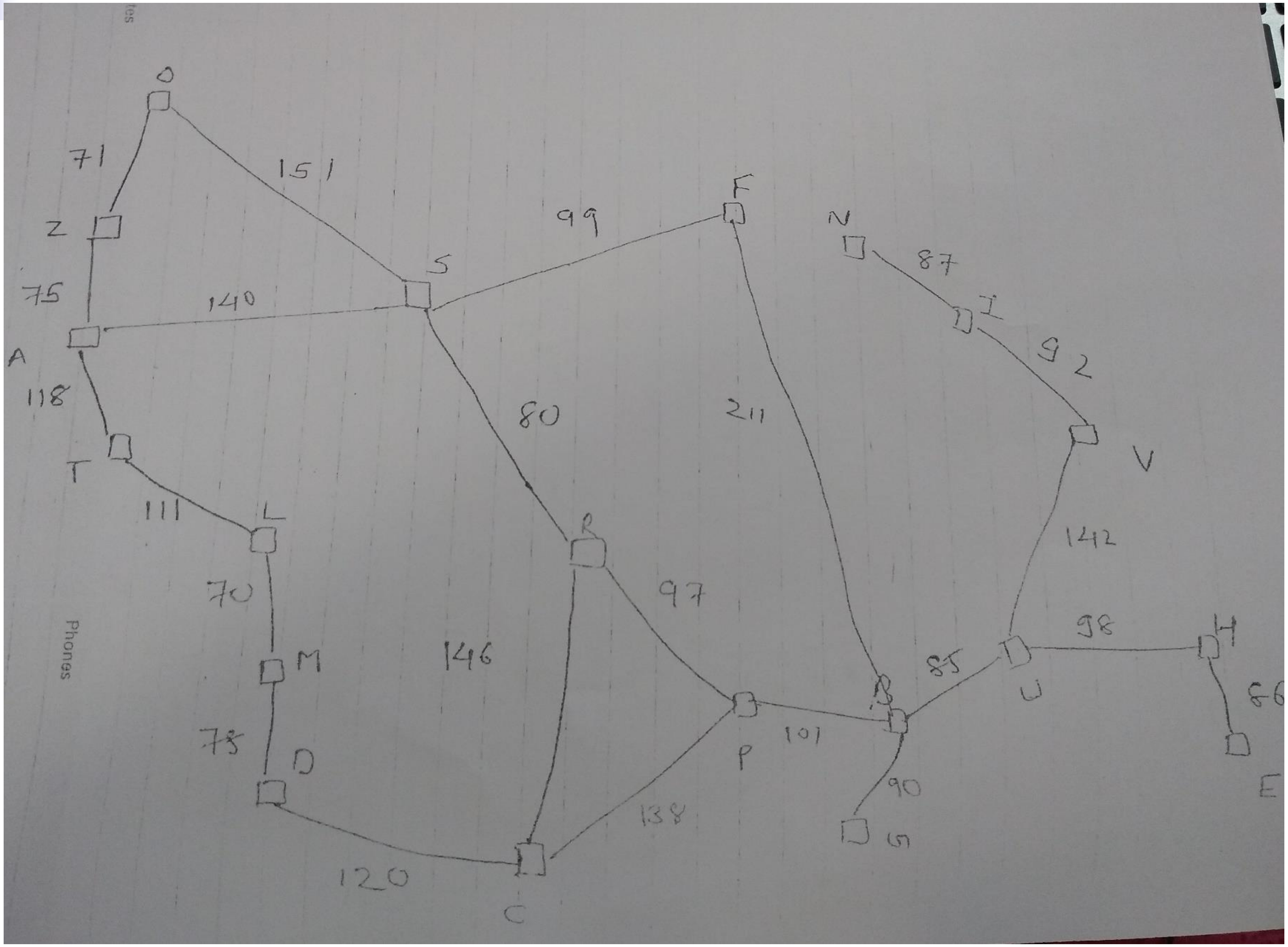


$$\begin{tikzcd}
 & B(8) & \\
A(8) \arrow[ur] \arrow[dr] & & \\
 & C(6) \arrow[loop above, "\text{already on closed}"] \arrow[r] \arrow[d] & D(5) \\
 & F(4) \arrow[r] & G(0)
\end{tikzcd}$$

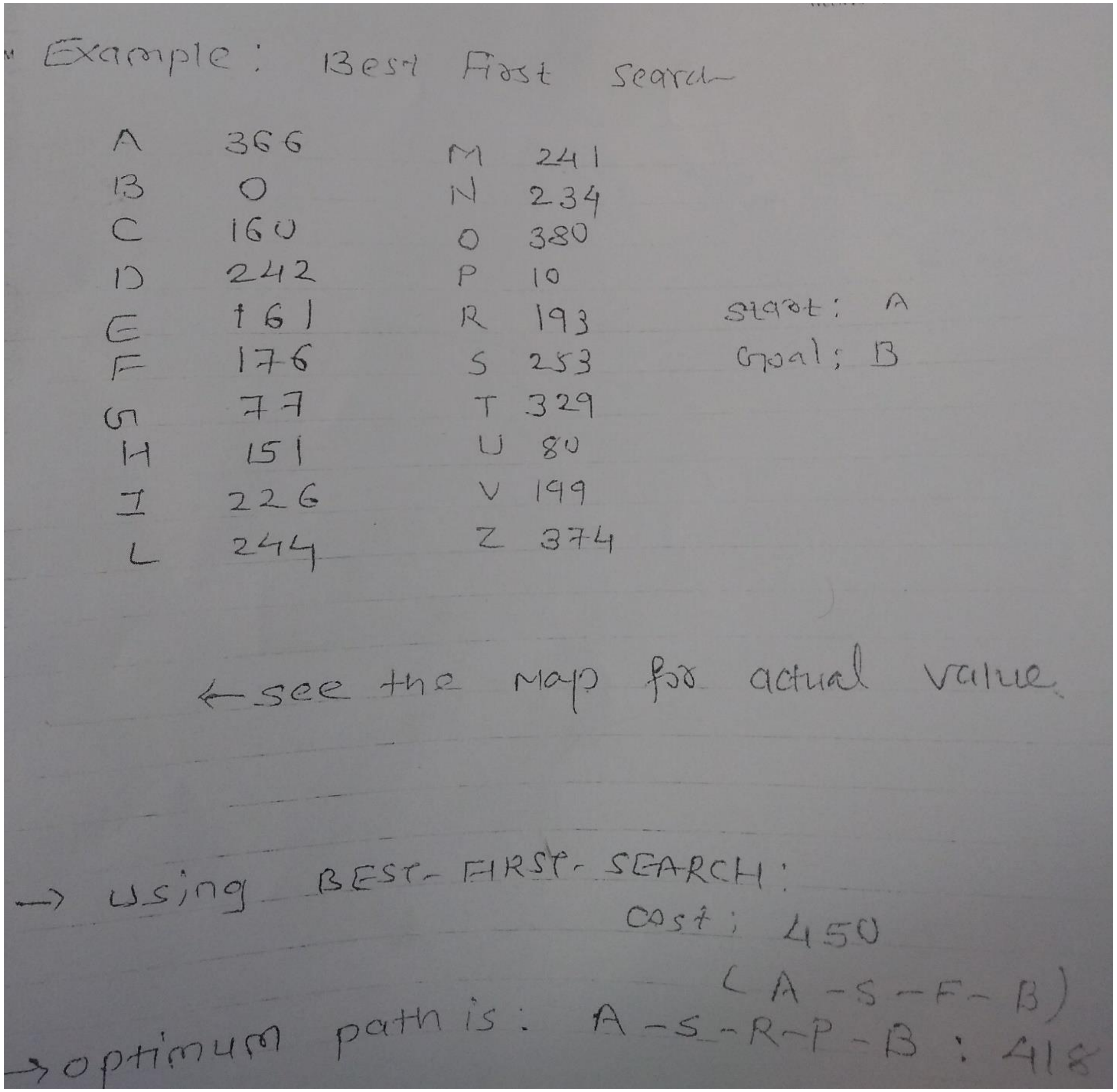
Path: $A \underset{1}{—} C \underset{6}{—} F \underset{8}{—} G = 15$ Actual cost

Optimal path: A – C – D – E – G with cost 'g'

Best-First Search Example2



Best-First Search Example2



Example : Best First Search

A	366	M	241
B	0	N	234
C	160	O	380
D	242	P	10
E	161	R	193
F	176	S	253
G	77	T	329
H	151	U	80
I	226	V	199
L	244	Z	374

start : A

Goal ; B

← see the map for actual value

→ using BEST-FIRST-SEARCH :

cost ; 450

(A - S - F - B)

→ optimum path is : A - S - R - P - B : 418

Beam Search

- Variation of Best-First Search. only the n most promising states are kept for future consideration
- Efficient for memory but there may be chances of missing a solution

A* algorithm

For the purpose of path finding two algorithms we have studied

I. Dijkstra's algorithm:

- Guarantees to find a shortest path from starting vertex using relaxation condition for positive weighted edge

  D[V] = min (D[V] , D[W] + C [W,V])

- It selects vertex closest to the starting points
- It is a slow algorithm

A* algorithm

II. Best First Search Algorithm

- Greedy in nature
- Does not guarantee to find a shortest path
- It selects the vertex closest to the goal
- It has some estimate (heuristic function value) of how far from the goal any vertex is
- It runs much quicker because of estimated value, which guides its way towards the goal

❑ A* combines heuristic approaches like greedy Best First Search and formal approach like Dijkstra’s algorithm

$$f' = g + h'$$