UCS521 MST Solutions

Q1: Consider a compiler for a machine with one register and set of instructions (described in Table 1). The first four instructions can only be executed when the register's contents are not divisible by 3 and when the register's contents are divisible by 3, the only instruction available is DIVIDE. The problem to consider is how to put a constant value in register assuming that the register begins with a value 1.

Consider the problem as a state space problem where the states of the problem are the contents of the register at any time. The operators (successor function) and the cost function are described in Table 1. Given the goal of having 9 in the register and the heuristic function h(n) = |9-n|. Find the solution of the problem using
(i) Best first Search
(ii) A* algorithm
Maintain OPEN and CLOSED at each step. Arrange the states with same evaluation function values in descending order of register contents.

Table 1: Successor and Cost Function

Instructions (Operators)	**Cost of Operators on the register containing number n**
ONE: Set the value of the register to 1	1
DOUBLE: Double the contents of the register	n
ADD: Add one to the contents of register	1
SUB: Subtract one from the contents of the register	1
DIVIDE: Divide the value of the register by 3	2n/3

Solution:

(i) BEST FIRST SEARCH **(//3 marks; 0.5 marks for each iteration)**

Let each state be represented as:

(current register content, parent register content, heuristic value of current)

Iteration	OPEN	CLOSED
0	(1,∅, 8)	-
1	(2,1,7) (0,1,9)	(1,∅, 8)
2	(4,2,5) (3,2,6) (0,1,9)	(1,∅, 8),(2,1,7)
3	(8,4,1)(5,4,4) (3,2,6) (0,1,9)	(1,∅, 8),(2,1,7) ,(4,2,5)
4	(9,8,0) (7,8,2) (5,4,4) (3,2,6) (16,8,7) (0,1,9)	(1,∅, 8),(2,1,7) ,(4,2,5) (8,4,1)
5	(7,8,2) (5,4,4) (3,2,6) (16,8,7) (0,1,9)	(1,∅, 8),(2,1,7) ,(4,2,5) (8,4,1) ,(9,8,0)

1→2→4→8→9

(i) A*SEARCH **(// 4 marks; 0.5 marks for iteration 0,1,2,5 and 1 marks for iteration 3,4)**

Let each state be represented as:

(current register content, parent register content, cost to reach current value + heuristic value of current)*

Iteration	OPEN	CLOSED
0	(1,∅,0+8)	-
1	(2,1,1+7) (0,1,1+9)	(1,∅,0+8)
2	(4,2,3+5) (3,2,2+6)(0,1,1+9)	(1,∅,0+8) (2,1,1+7)
3	(8,4,7+1) (5,4,4+4)(3,2,2+6)(0,1,1+9)	(1,∅,0+8) (2,1,1+7) (4,2,3+5)
4	(9,8,8+0) (5,4,4+4)(3,2,2+6)(7,8,8+2) (0,1,1+9)(16,8,8+7)	(1,∅,0+8) (2,1,1+7) (4,2,3+5)(8,4,7+1)
5	(5,4,4+4)(3,2,2+6)(7,8,8+2)(0,1,1+9) (16,8,15+7)	(1,∅,0+8) (2,1,1+7) (4,2,3+5)(8,4,7+1) (9,8,8+0)

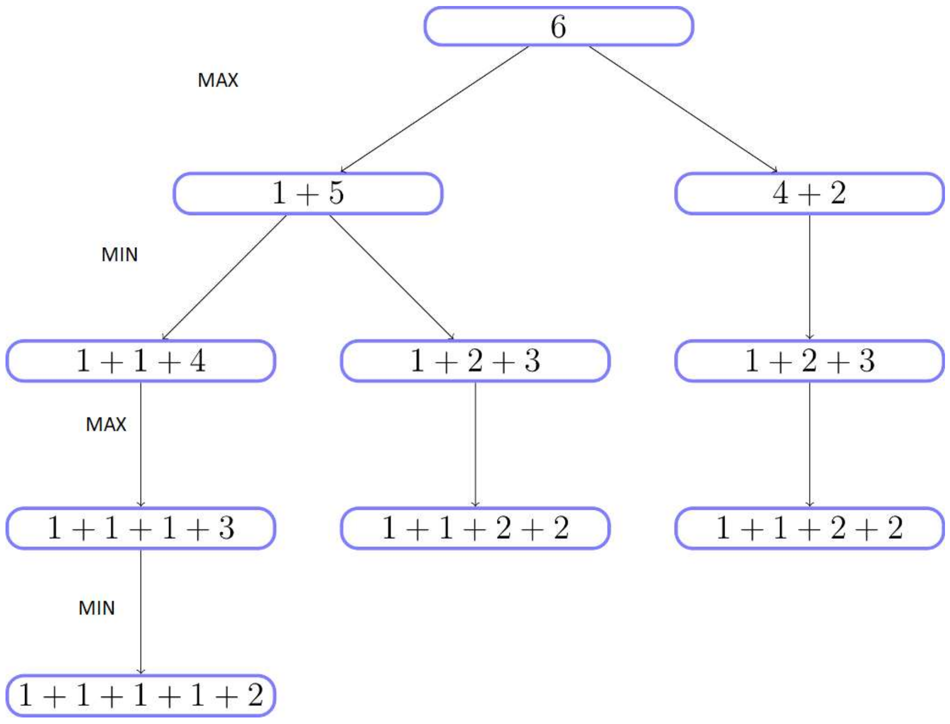
1→2→4→8→9 with path cost 8

Iteration I:
Successors of 1:
Through ONE (1): newg(1)=g(1)+cost(1,1)=0+1 =1
Not less than g(1) . Reject it
Through DOUBLE/ADD (2) : g(2) =g(1) +cost(1,2) =0+1=1
f(2)=1+7
Through SUB (0): g(0)=g(1)+cost(1,0)=0+1=1
f(0)=1+9
Iteration II:
Successors of 2:
Through ONE (1) : newg(1)=g(2) +cost (2,1)= 1+1=2
Not less than g(1) . Reject it
Through DOUBLE (4): g(4) =g(2) +cost (2,4) = 1+2=3
f(4)=3+5
Through ADD (3): g(3)=g(2) + cost (2,3)=1+1=2
f(3)=2+6
Through SUB (1): newg(1)=g(2)+cost(2,1) = 1+1=2
Not less than g(1) . Reject it
Iteration III:
Successors of 4:
Through ONE (1): newg(1)=g(4) +cost(4,1)= 3+1=4
Not less than g(1) . Reject it
Through DOUBLE (8) : g(8)=g(4) +cost (4,8) =3+4=7
*f(8)= 7+1
Through ADD (5): g(5)=g(4) +cost(4,5)=3+1=4
f(5)=4+4
Through SUB (3): newg(3)=g(4) +cost(4,3)=3+1 =4
Not less than g(3) . Reject it
Iteration IV:
Successors of 8:
Through ONE (1): newg(1)=g(8) +cost(8,1)= 7+1=8
Not less than g(1) . Reject it

Through DOUBLE (16) : g(16)=g(8) +cost (8,16) =7+8=15
f(16)= 8+7
Through ADD (9): g(9)=g(8) +cost(8,9)=7+1=8
f(9)=8+0
Through SUB (7):g(7)=g(8) +cost(8,7)=7+1 =8
f(7)= 8+2

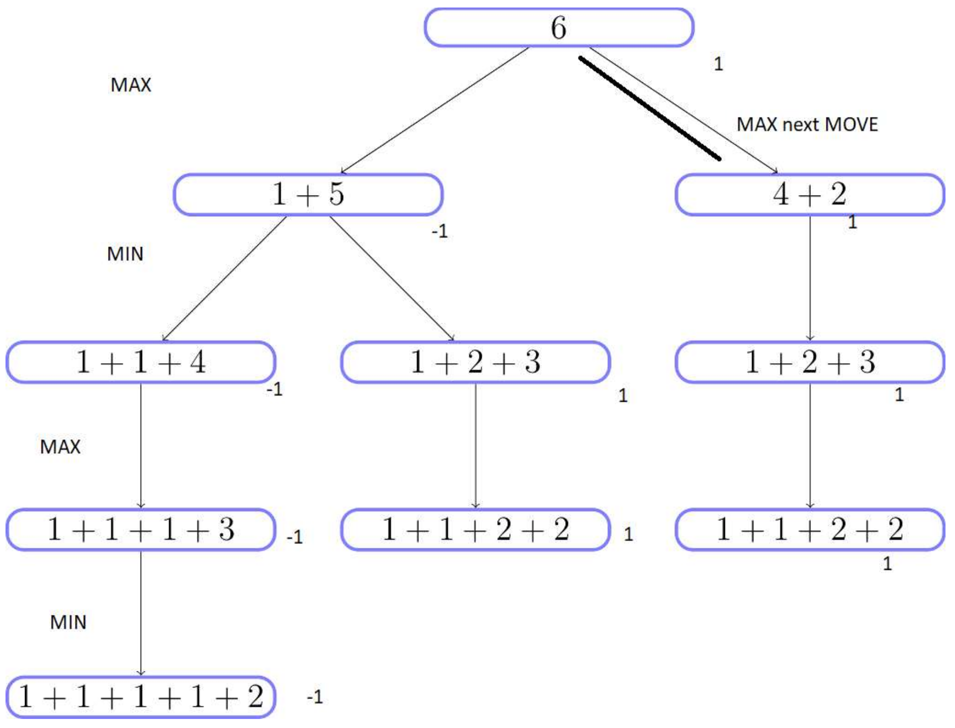
Q2: Consider a robotic arm that can perform five actions namely REACH, SLIDE, PUSH, PICK, PLACE. The agent architecture has 32- bit storage. How many agent functions are possible for the given robotic arm agent?
Solution: **(//2 marks ; 1 mark for total percepts and 1 mark for total agent functions)**
Total number of percepts (states) that can be stored = 2^{32}
Each state can be filled with any one of the 5 actions i.e. in 5 ways
Total agent functions possible = 5×5×5………………..2^{32} times = $5^{2^{32}}$

Q3: Consider a Nim game in which initially there is a single pile of *n* tokens placed between two opponents. The opponents play alternately and at each turn the player selects a pile and divides it into two piles of unequal number of tokens. The first player who is unable to make a move loses the game.
For instance, a pile of 8 tokens may be divided into two piles 7 and 1, 6 and 2, 5 and 3 but not 4 and 4. In the next move , another player can further divide the two piles containing 7 and 1 tokens into three piles containing 6,1,1 tokens or 5,2,1 tokens or 4,3,1 tokens.
(i) Generate the game tree for n=6.
(ii) If MAX starts the game, find the next move of MAX using Minimax algorithm.
Solution:
(i) Game Tree **(// 3 marks)**



(ii) Minimax Algo **(//2 marks)**

$$\text{Let } value(leaf) = \begin{cases} 1\ if\ MAX\ winning\ positon \\ -1\ if\ MIN\ winning\ position \end{cases}$$



Q4: Consider a finite search tree of depth d and branching factor b. Suppose the shallowest goal node is at depth g ≤ d. What is the minimum and maximum number of nodes that might be generated by a breadth-first search?

Solution: (//2 marks; 1 mark for minimum nodes and 1 mark for maximum nodes)

Minimum number of nodes =1 (goal node lies at the root of the tree)

Maximum number of nodes

The goal node is the last node at depth d

Total nodes generated =$1+b+b^2+b^3+$.................................$b^d = \frac{(b^{d+1}-1)}{(b-1)}$

Q5: Prove the following properties of heuristics for the A* algorithm:

(i) If h_1 and h_2 are two admissible heuristics such that $h_1 \geq h_2$ then number of nodes expanded by first algorithm (which uses heuristic h_1) are also expanded by the second algorithm (which uses heuristic h_2).

(ii) A monotonic heuristic is always admissible i.e. it always underestimates the optimal values.

*Do not prove the properties by taking specific examples.

Solution:

(i) **(// 3marks; 0.5 marks for step 1 and 2 and 2 marks for step 3)**

The property is used by PMI

Step 1: for depth =1

Start node S is expanded by every A* algorithm irrespective of the heuristic function used

Therefore start node S expanded by A1* is also expanded by A2*
Step 2: Let it be true for depth k i.e. all nodes expanded by A1* up to depth k are also expanded by A2*
Step 3: To prove: Any node n expanded by A1* at depth (k+1) is also expanded by A2*
Proof by contradiction
Assume: Let node n is expanded by A1* but A2* terminates without expanding n
Since, n is expanded by A1*
Therefore, $f1(n) \leq f^*(G)$
$g1(n) + h1(n) \leq f^*(G$
$h1(n) \leq f^*(G) - g1(n)$……………(1)
Since n is not expanded by A2*
Therefore, $f2(n) \geq f^*(G)$ [Otherwise it would have been expanded by A1*]
$g2(n) + h2(n) \geq f^*(G)$
$g1(n) + h2(n) \geq f^*(G)$
[Because $g1(n) \geq g2(n)$ as all nodes seen up to depth k by A1* are also seen by A2*. Thus A2* will find a better or equivalent path to n at depth k+1]
$h2(n) \geq f^*(G) - g2(n)$……………(2)
Combining (1) and(2) $h1(n) \leq h2(n)$
But $h1(n) > h2(n)$
Therefore our assumption is wrong and any node n seen by A1* at depth k+1 is also expanded by A2*

(ii) **(//3 marks)**
Let S-A-B-C-D…………N-G is the optimal path using A* algorithm which uses monotonic function.
Since the heuristic function is monotone, therefore, the heuristic values of each segment is underestimated

i.e. $h(S)-h(A) \leq \text{cost }(S,A)$
Similarly, $h(A)-h(B) \leq \text{cost }(A,B)$
$h(B)-h(C) \leq \text{cost }(B,C)$
$h(C)-h(D) \leq \text{cost }(C,D)$
.
.
.
$h(N)-h(G) \leq \text{cost }(N,G)$

Adding all the equations
$$h(S)-h(G) \leq \text{cost}(S,A) + \text{cost }(A,B) + \text{cost}(B,C) + \text{cost }(C,D) + \text{cost }(N,G)$$
Since $h(G) = 0$
$$h(S) \leq \text{optimal cost }(S,G)$$
The same holds true for all other nodes

Q6: Consider the problem of decoding a secret word 'BANANA'. The probability of correctly identifying the secret word is quite less (specifically it is $1/(26)^6$ i.e. less than 1 in 308 million). Simulate this combinatorial explosive problem with Genetic Algorithm and output first two iterations.
Consider population size of 4 where each chromosome is represented as a six letter word ($X_1X_2X_3X_4X_5X_6$) where $Xi \in \{A,B,C…Z\}$. Let the initial population be BAHAMA, ABCDEF, IJKLMN, and CABANA. Generate the population for next iterations as follows:

Select the 1st and 2nd fittest individual as it is in the next iteration.
Apply 1-point crossover in the middle between 3rd and 4th fittest chromosome.
Apply single letter mutation of first offspring produced through crossover between 3rd and 4th fittest chromosome. Letter chosen for mutation follows this cyclic order {X_6,X_5,X_4,X_3,X_2,X_1}. The letter to be mutated must be shifted two letters down i.e. A must be replaced with C and so on.
Use the following fitness function:

$$Fitness(Chromosome) = \sum_{n=1}^{6} |Guess(n) - Answer\ (n)|$$

Where Guess (n) and Answer (n) are the ASCII values of the letters at nth position in the guessed chromosome and the answer. Given that ASCII value of A and Z are 65 and 90.

Solution: **(// 5 marks ; 2.5 marks for each iteration; 1 marks for selection; 1 for crossover and 0.5 for mutation)**
Iteration 1:

Chromosomes	Fitness values	Crossover	Mutation
BAHAMA	0+0+6+0+1+0=7		
ABCDEF	1+1+11+3+9+5=30	ABC | DEF = ABCLMN	ABCLMP
IJKLMN	7+9+3+11+1+3=44	IJK | LMN IJKDEF	
CABANA	1+0+12+0+0+0=13		

Iteration II

Chromosomes	Fitness values	Crossover	Mutation
BAHAMA	0+0+6+0+1+0=7		
ABCLMP	1+1+11+11+1+15= 40	IJK | DEF = IJKLMP	IJKLOP
IJKDEF	7+9+3+3+9+5=36	ABC | LMP ABCDEF	
CABANA	1+0+12+0+0+0=13		

Population for next iteration is BAHAMA, IJKLOP, ABCDEF, CABANA

Q7: Consider ring and hook problem in which there are few rings in water which are to be put in hooks. Initially we randomly rotate the game to put as many rings as possible in the hooks. But when we have put larger number of rings, we rotate carefully so that rings which are in hooks do not come out.

Name the searching algorithm which works analogous to the solution strategy of ring and hook problem? Explain how such behaviour is controlled for the search algorithm?
Solution: **(3 marks; 1 marks for name and 2 marks for explanation)**

Simulated Annealing

The temperature and change in objective function value controls the behaviour of simulated annealing algorithm. The algorithm starts of being closer to the Random Walk (at high initial values of temperature), but gradually becomes more and more controlled controlled by the gradient gradient and become more like Steepest Hill Climbing (as the temperature is lowered). The probability of making a move at any point of time is given by a sigmoid function of the gain ΔE and temperature T as given below:

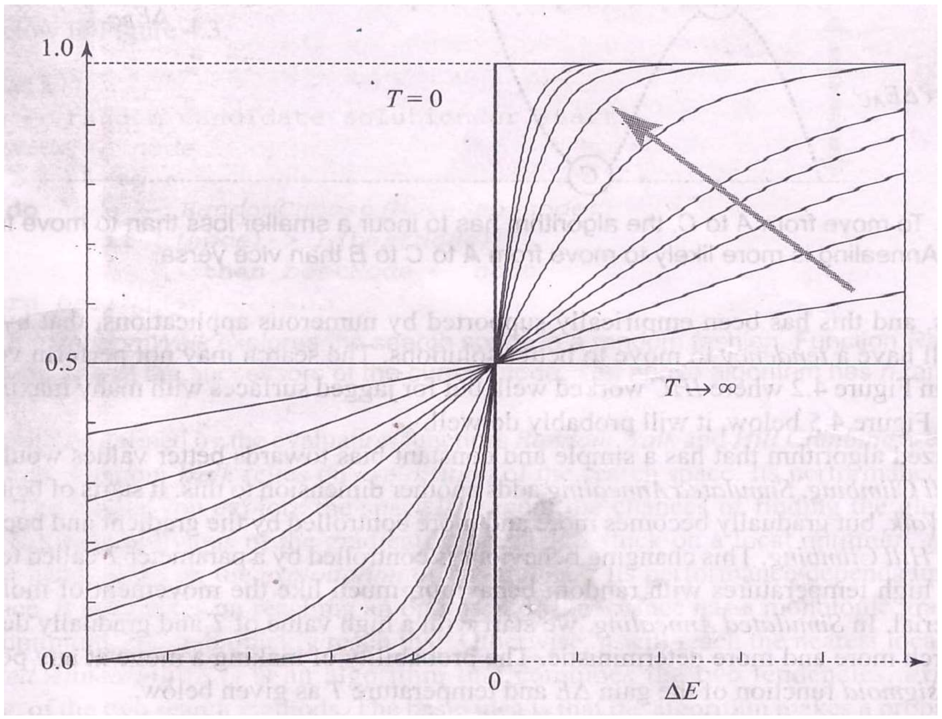
$$P(c,n) = \frac{1}{1 + e^{-\Delta E/T}}$$

(for maximization)

$$P(c,n) = \frac{1}{1 + e^{\Delta E/T}}$$

(for minimization)

The probability as a function of the two values is shown below. The Y-axis shows the probability values, and the X axis varies ΔE. The different plots are for different value of T.
When the temperature tends to infinity, the probability is uniformly 0.5, irrespective of ΔE. This means that there is equal probability of accepting a good move or bad move. This is like *Random Walk* algorithm.
Another observation from the graph is when ΔE =0, the probability is 0.5. This means that when the next node evaluates to same as current then it is equally likely to accept the next node.
Further, when the temperature is lowered, the probability of taking good move keeps on increasing from 0.5 and the probability of bad move keeps on decreasing from 0.5.
At very low values of temperature, the sigmoid probability function tends to become more and more step function.
Finally at T=0, it is a step function and the decision to move to neighbor becomes deterministic. The search moves to the neighbor (with probability 1) for good move (ΔE>0) and does not move to it (with probability 0) for bad move (ΔE<0).This is like the Hill Climbing algorithm


$T = 0$
$T \to \infty$
ΔE