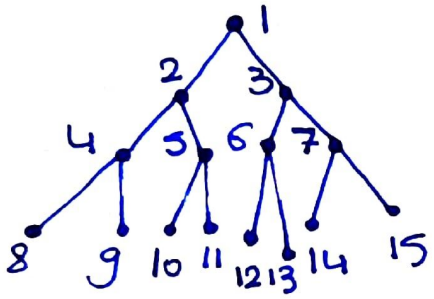
MI Sheet 3

1. Consider a State Space where
   → Initial State is 1
   → A State K has two successors 2K & 2K+1

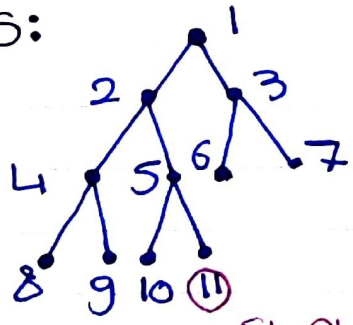
a) Draw a Portion of the State Space (for States 1 to 15)



- Its a tree

b) let 11 be goal State

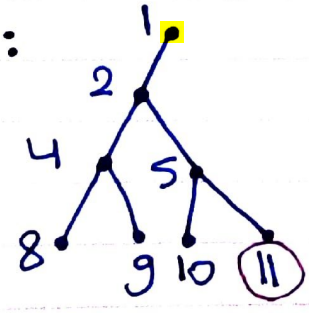
BFS:



Path: 1 → 2 → 5 → 11

Stop! (once 11 was inserted)

DFS:



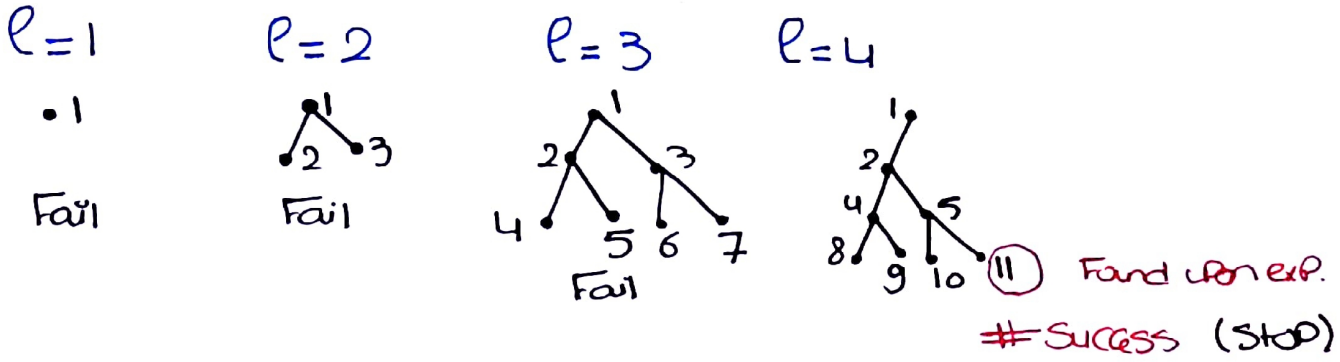
// depth = 0

Path: 1 → 2 → 5 → 11

Stop! (once 11 was expanded)

Depth-limit with d = 3:
→ treat nodes at depth 3 as if they have no succ.
- They already don't (eq. to DFS above)

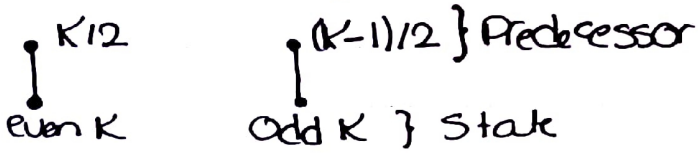
Iterative Deepening



C. How well would bidirectional search work for this Problem what's the branching factor along each direction?

→ To apply bidirectional search, we need to be able to compute Predecessors of a State.

here its easy



and its clear at this Point that the branching factor along this direction is 1

- It would work well (reduce BFS complexity from $O(b^d)$ to $O(b^{d/2})$ as we'd expect. ①

Frontiers

From S=1	From G=11
[1]	[11]
[2, 3]	[5]
[3, 4, 5]	STOP! Frontiers intersect
1→2→5	←Path→ 11←5

1→2→5→11

→ Due to the smaller branching factor, searching backward is even easier.

d) Reformulate the search problem so it can be solved with almost no search.

→ Just do backward BFS

Initial State: 1 1 (or goal n in general)

Goal test: State = 1

Action(S): only one action at any State S which is go to Predecessor

Successor function: Result(S,a) = $\lfloor S/2 \rfloor$

i.e., K even → K/2
K odd → (K-1)/2

Path Cost: Sum of Steps along Path, each Step Costs 1

e) Find the search algo that solves d

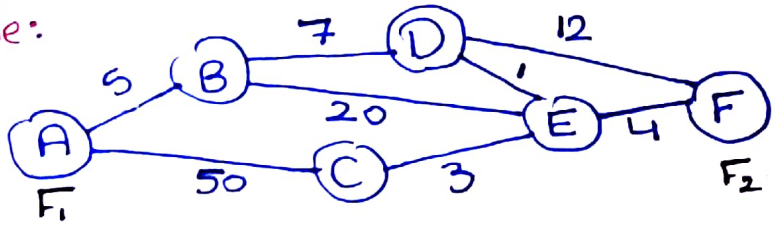
→ Just BFS

append

- If the State we're at is even Push 'left' into list else Push 'right' before moving to next State (Predecessor)
- the reversed list at end of algo. has our Solution

3.3) → Two Friends live in different Cities on a map
→ on every turn can simult. move each friend to a neighboring City i,j
→ any two neighboring Cities have distance d(i,j) between them
→ the time it takes for one of the Friends to move to next City (from i to j) is also d(i,j)
→ when a Friend arrives to next City before the other it must wait for them
• We want the two Friends to meet as quickly as Possible.

Example:



• F1, F2 are the Friends

• Possible traversal:

F1	B	D
F2	E	D
Cost	12	7

// want to min. total cost

a) Formulate

Initial State: (A, E) where A,E are the two Cities they're initially in

Goal: State = (I, I) For some City I

Action: The Possible actions from State (x,y) is move(x`,y`) for each x`,y` that is a neighbor of x,y respectively.

Successor function:

- move (x', y') updates the state to (x', y') given that it was (x, y)
  i.e., the successor of state (x, y) after doing move(x', y') is (x', y')

Path cost:

- Is the sum of stepCost
- Step cost from $S = (x, y)$ to $S' = (x', y')$ is $\max(d(x, x'), d(y, y'))$ where $d(.,.)$ is the distance between the two neighboring cities given

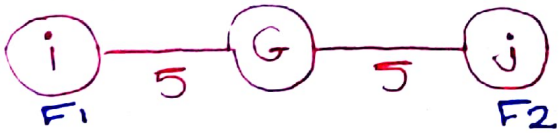
b) let $D(i, j)$ be the St. line dist. between cities i, j which of the following is admissable.

underestimates cost to goal.

G
d

→ $D(i, j)$

- Not admissable
- Consider



- The cost to goal is $\max(5, 5) = 5$
- Suppose the St. line dist is equal to road dist. then $\underbrace{D(i, j)}_{10} \geq \underbrace{h^* = 5}_{\text{true cost to goal}}$

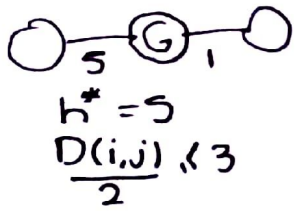
→ $\frac{D(i,j)}{2}$

- This time even in the worst case scenario where road dist. = euclidean distance and both cities are equally apart from a 3rd one

→ $\frac{D(i,j)}{2} = h^* = 5$

↳ in general will be even smaller as

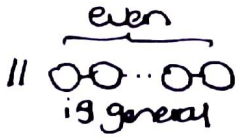
1. road dist $\geq$ St. line dist.
2. Cities may be not equally far apart



. It's thus admissable

→ $2D(i,j)$
Not admissable as we've found
$2D(i,j) \geq D(i,j) \geq h^*$

C) are there completely connected maps for which there's no solution



No sequence of the available actions lead the two friends to reach the same state.
- the action requires that both move simultaneously

So, only possibilities here are
$S = (A,B) \longrightarrow S' = (B,A)$
$S = (B,A) \longrightarrow S' = (A,B)$
and none is a solution

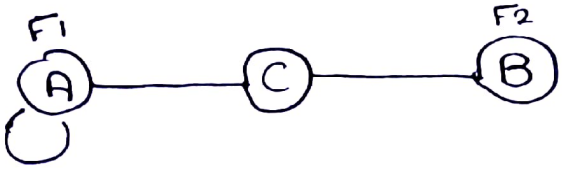
. $A \neq B$ ∴

d) Are there maps where
→ All solutions require one friend to visit same city twice.
- we want it to apply for all solutions
→ let's start with a map with no solution

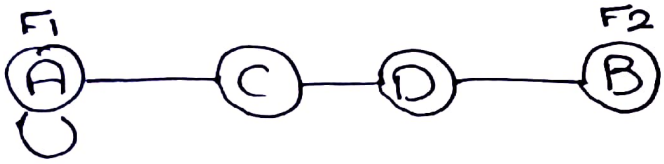
- we solve the problem if there's a way for one of the friends to stay in the city (or waste time around)

- Consider a self loop (road to the same city)

- F2 goes to A & F1 goes to A via loop
- but we want A to be visited twice



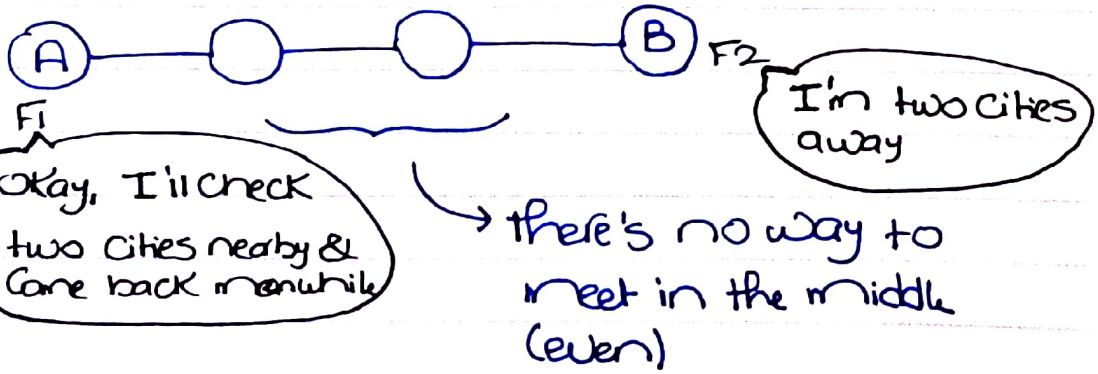
- but now the sol. (A,B) → (C,C) is also possible

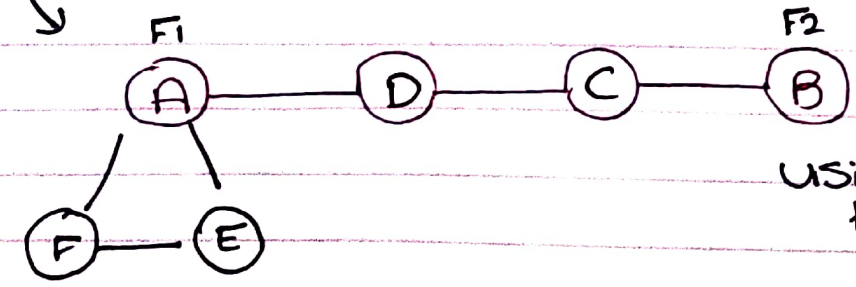


- The only possible sol. is to meet at A or C
  - → For C, F1 visits A twice
  - → For A, it does so thrice

- Don't think we can do better with self-loops

Consider





The only sol. here: (A, B)

For E, C

E or F, D

(A, A)

visited twice

3.14)

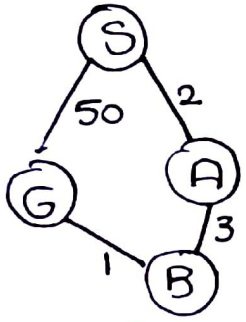
⇒ which of the following is true/false & explain

a) DFS (always) expands (at least) as much nodes as A* with an admissable heuristic.

i.e., if N is the nodes expanded

$N_{DFS} \geq N_{A^*}$

False. Consider



→ DFS goes to goal directly
→ Admissable A* is optimal and will choose longer path in terms of nodes.

- there are still cases where A* will expand less depending on where's the goal

b) $h(n) = 0$ is admissable for 8-Puzzle

For any Prob.

True, its always admissable as $h \leq h^*$ will be always true.

↓ 0 ↓ true cost to goal can't be less than 0

0 (we always assume +ve step costs)

c) Can't use A* when state space is continuous

- False

→ just discretize it with small enough error

(e.g. $x \in [1,10] \rightarrow x \in \{1, 1.01, 1.02, \ldots, 2, 2.01, \ldots, 10\}$

d) BFS is Complete even if StepCost can be 0

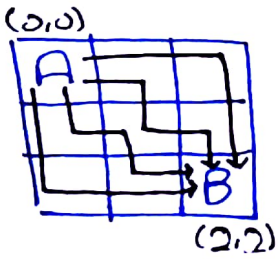
- True
- It keeps exploring until goal is found or all nodes are explored

→ all we need is a goal at finite depth for Comp.

e) Consider the Problem of moving ♖ (Rook) from square A to B on chessboard in least no. of moves

→ is manhattan distance an admissable heuristic

- False



it takes at least 2 moves ↳ or ↳ but manhattan dist is $|2-0| + |2-0|$

- Can also consider

manh. = 8 > moves = 1

3.29)

→ Prove that any Consistent heuristic is admissable

- Consistency $h(n) - h(n') \leq c(n, n')$
- Admissability $h(n) \leq h(n^*)$

⇒

- let the Path from $\underbrace{n}_{n_0}$ to $\underbrace{G}_{n_k}$ involve nodes $n_1, n_2, \ldots, n_{k-1}$
- then $h(n^*) = \sum_{i=0}^{k-1} c(n_i, n_{i+1})$

→ We know that

$$h(n_0) - h(n_1) \leq c(n_0, n_1)$$

$$h(n_1) - h(n_2) \leq c(n_1, n_2)$$

$$\dots$$

$$h(n_{k-1}) - h(n_k) \leq c(n_{k-1}, n_k)$$

by adding all

$$h(n_0) - \underbrace{h(n_k)}_{\text{zero}} \leq \sum_{i=0}^{k-1} c(n_i, n_{i+1})$$

(n_0: start, n_k: goal)

$\Rightarrow$ Thus, $h(n) \leq h(n^*)$

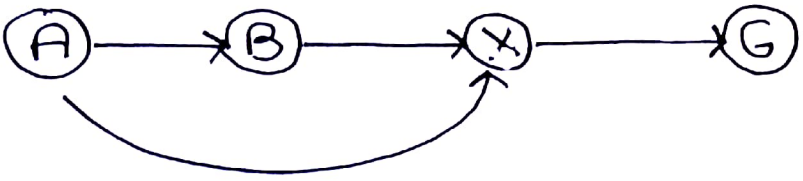
- Of course this holds for any n by universal generalization (we proved for arbitrary n) □

→ Construct admissable heuristic that is not consistent. (also Q 3.24)

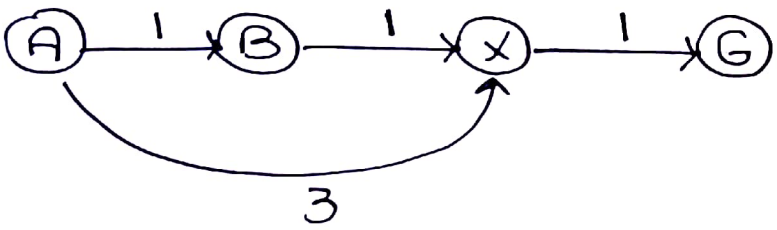
- Recall that this is the case where graph search fails
  - → It occurred when a node X is found at a better cost after its explored
    - Graph search can't go back and take it due to explored set
    - → So we need a node X reachable from 2 paths

- The way we set the admissable heuristic should cause us to expand X from the less optimal path.

→ The simplest graph satisfying these (where X isn't goal)



- let the horizontal path be more optimal



- most basic setting of costs
  → 1,1,1
  → now need 3

- Now we want the algo. to expand X right away

$1 + h(B) > 3 + h(X)$

→ most value for $h(B)$ is 2
→ least value for $h(X)$ is 0
→ can't satisfy with current costs
  - let's try to change $C(X,G)$ so we can choose higher $h(B)$
  - or lets just set it as variable and solve for it

→ $1 + \underbrace{h(B)}_{1 + C(X,G)} > C(A,X)$ // no longer held at 3

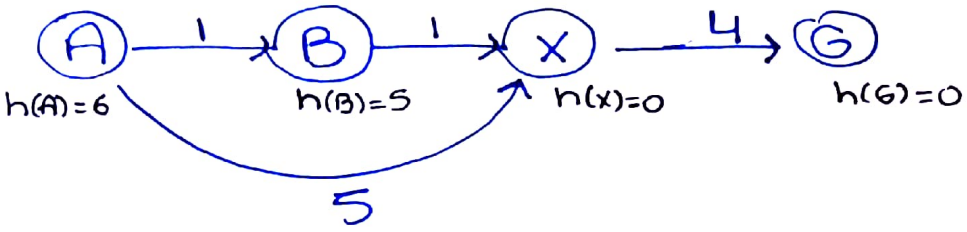
→ $1 + 1 + C(X,G) < C(A,X) + C(X,G)$

The two eqns give us

$2 > C(A,X) - C(X,G)$
$2 < C(A,X)$

- take $C(A,X) = 5$ and $C(X,G) = 4$

→ $h(B) = 1 + C(X,G) = 5$



- Here graph search clearly fails
$(A \to X \to G)$
⇒ Cost = 9 > 6

- h is clearly admissable
- h is not consistent $h(B) - h(X) \leq C(B,X)$ doesn't hold

3.25)

→ Consider best-first search
→ Using the evaluation function (decides Priority queue order):

admissable

$$f(n) = (2-w)g(n) + wh(n)$$

- At $w = 0$
$f(n) = 2g(n)$
So it's UCS

// multiplying by 2 → same Priority Q order

→ Optimal & Complete

- At $w = 1$
$f(n) = g(n) + h(n)$
So it's A*

→ Optimal (tree search) & Complete

- At $w=2$
  $f(n) = 2h(n)$
  → Not optimal & complete (graph search)

- All of these are justified in the lec.

⇒ For any w
→ As long as the state space is finite (in depth & branching) then graph search will be complete
- Even if frontier is ordered randomly, we eventually explore the whole graph.

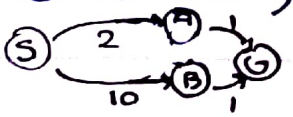
- For $w > 2$ (Following wasn't done in tutorial)

$$f(n) = \underbrace{(2-w)}_{-ve} g(n) + \underbrace{w}_{+ve} h(n)$$

$$= bh(n) - ag(n) \qquad (a, b \text{ are } +ve)$$

- To show that this isn't guaranteed to be optimal
  → let $h(n) = 0$ (still admissable)

- Now it expands the node for which the path is largest (obviously not optimal; won't find shortest path).



- For $w < 0$

$$f(n) = ag(n) - bh(n) \qquad (a, b \text{ are } +ve)$$

At the goal state

$$f(G) = ag(G)$$

So even if $g(G) < g(n) + h(n)$ (goal should be expanded first) we easily have $ag(G) > ag(n) - b\,h(n)$ as the two terms are now being subtracted.

→ hence, generally not optimal as well.

3.21)

• Prove the following or give counterexample

a) BFS is a special case of UCS

→ True.
→ Let all edges have the same cost
  • then nodes will be ordered by depth
  • Shallowest node will be expanded first

b) DFS is a special case of best-first

→ True
→ let the heuristic be $\underbrace{h(n)}$ $-\text{depth}(n)$
  • Now we expand deepest node first

c) UCS is a special case of A*

→ True
→ let $h(n) = 0$
  • Now we expand node of lowest cumulative path cost first ($g(n)$)