# Assignment 1.

Q1. Write a dynamic programming algorithm for creating an optimal binary search tree for a set of n keys. use the same algorithm to construct the optimal binary search tree from following.

| Key | A | B | C | D |
|---|---|---|---|---|
| probability | 0.1 | 0.2 | 0.4 | 0.3 |

→ The optimal binary tree problem involves constructing a binary search tree for a given set of keys will be generated using

Steps -

1. Input -

We have n keys with given probabilities for accessing each key. Here the keys are A, B, C, D with their respective probabilities P(i)

we ~~ha~~ also need dummy key q(i) which represent unsuccessful searches.

2. Cost Calculation -

We need to compute the cost of searching the tree. For each subtree the cost includes the sum of the probabilies of the all keys in the

3. Dynamic Programming table.

Use a 2D table e[i][j] where e[i][j] represents the expected cost of the optimal BST for keys from i to j.

Another table w[i][j] is used to store sum of probabilities p[i] to p[j].

A root [i][j] keep track of root of optimal BST.

4. Recurrence Relation.

base case for single key

e[i][i] = p[i] , and for no key, e[i][i-1] = q(i-1)

for more than one key

$e[i][j] = \min_{r+1} (e[i][r-1] + e[r+1][j] + w[i][j])$

~~Using the~~

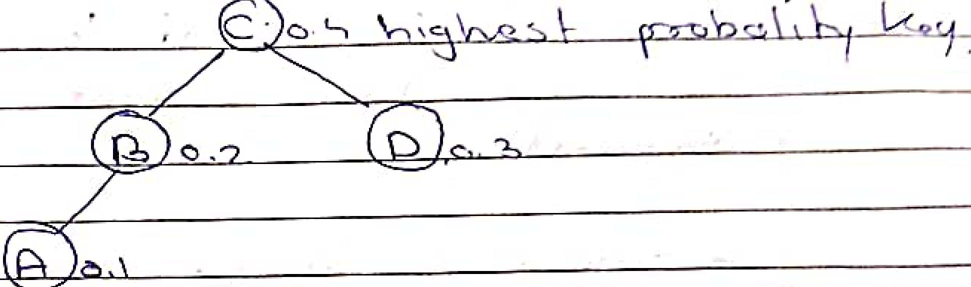
5. Compute Optimal Cost and structure.

Using the recurrence relation, compute the cost for each possible range of keys and choose the root that gives the minimum cost.

6. Construct the Optimal Tree

use root table to reconstruct the tree by choosing the root at each step.

For given data using dynamic programming algorithm structure of tree might look like.




This structure minimizes the expected search cost as higher probability key are closer to root

Q2 What is the branch and bound method? Write control abstraction for lest cost search

* Branch & Bound Method-

The branch and bound method is used for solving optimized problems when dealing with combinatorial search space like traveling salesman problem or integer programming. It systematically explores all possible solutions by dividing problem and calculating a bound on best possible solution in each division.

Steps of branch and bound.

1 Branching - split the problem into smaller subproblems

2. Bounding - for each subproblem compute a bound on the best possible solution that can be achieved within that subproblem. if bound of subproblem is worse than current best solution discard it

3. selection - choose the most promising subproblem to explore based on bounds

4. Pruning - if a subproblem's bound is worse than the best-known solution eliminate it from consideration

* Control Abstraction for least cost search

- Initialize the search with starting node
- Expand nodes by generating all children.
- For each child, calculate the cost and compare it with the current best solution
- if the cost of child node is better update best solution.
- Prune braches that cannot produce a better solution
- Repeat until all nodes are either explored or pruned

Control Abstraction Example.

```
function branch & bond (problem):
    initialize priority queue with intial state
    best solution = infinity
    While priority queue is not empty
        node = priority_queue.pop ()
        if (node is a solution:
            if cost(node) < best solo
                bestsol = cost (node)
            else
                for child in expand (node):
                    if bound child) < best sol
                        priority_queue.push
                                (child)
    return best solution
```

In this abstraction priority queue is used to keep track of nodes to explore & we prune suboptimal nodes using bound function