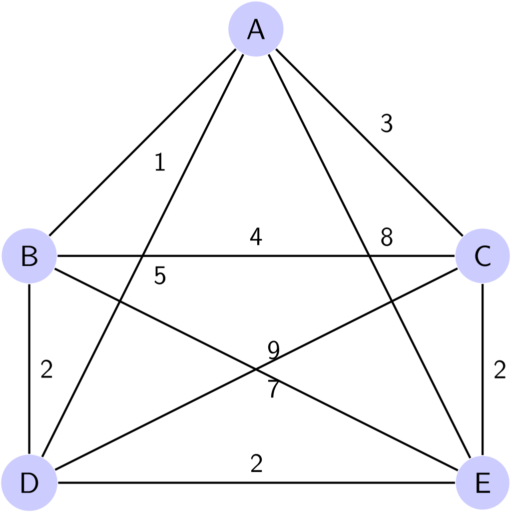
Traveling Salesman Problem



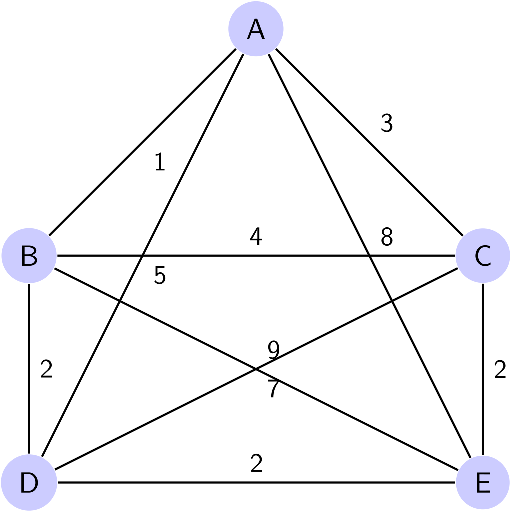
Review

- Greedy, 2-Opt
  - Fast
  - Approximate
  - Simple
- Dynamic Programming, Held-Karp
  - Slow
  - Memory intensive
  - Optimal
- Randomized, Genetic
  - Fast
  - Approximate
  - Tuning necessary

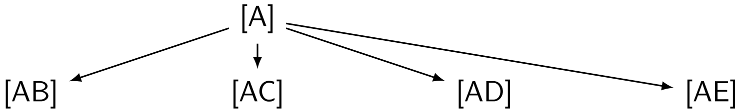
Backtracking for Optimization?

- Find any solution with DFS
- Throw out partial solutions if they are worse than the solution
- Update the solution if a better one is found

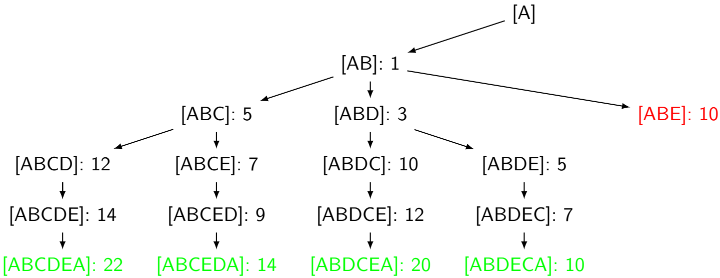
Example Graph



State Space

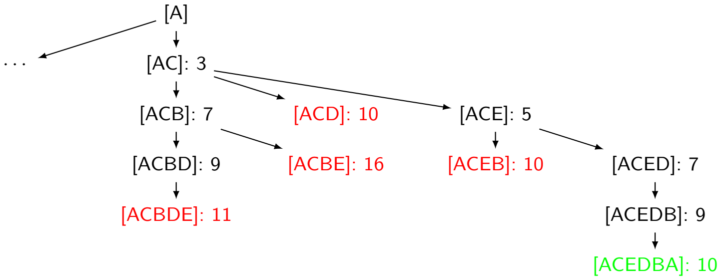


Example

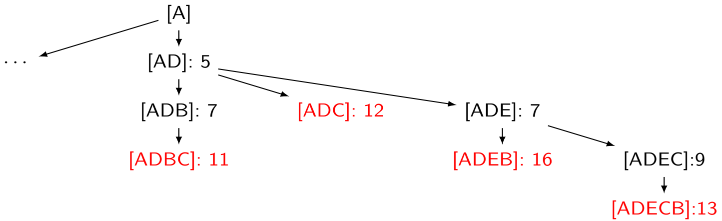


[A]
[AB]: 1
[ABC]: 5
[ABD]: 3
[ABE]: 10
[ABCD]: 12
[ABCE]: 7
[ABDC]: 10
[ABDE]: 5
[ABCDE]: 14
[ABCED]: 9
[ABDCE]: 12
[ABDEC]: 7
[ABCDEA]: 22
[ABCEDA]: 14
[ABDCEA]: 20
[ABDECA]: 10

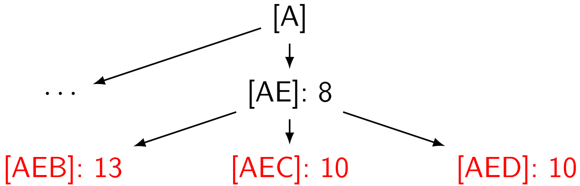
Example


[A]
...
[AC]: 3
[ACB]: 7
[ACD]: 10
[ACE]: 5
[ACBD]: 9
[ACBE]: 16
[ACEB]: 10
[ACED]: 7
[ACBDE]: 11
[ACEDB]: 9
[ACEDBA]: 10

Example



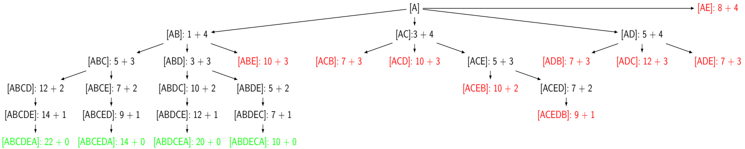
Example



Heuristics

- Guess remaining costs
  - $dist([ABDC]) = 1 + 2 + 9$
  - What about connecting E and A?
- Admissible - never overestimate remaining costs

Heuristic Example



Statistics

Method	**Node Count**
Exhaustive	64
Branch & Bound	39
Branch & Bound + Heuristic	25

Pseudo Code

```
procedure BranchBound(Graph graph, Vertex depot)
    stack = empty stack
    best = ∅
    stack.push([depot])
    minCost = ∞

    while stack not empty do
        current = stack.pop()
        if pathLength(current) + heuristic(current, graph) < minCost then
            if isComplete(current) then
                best = current
                minCost = pathLength(current)
            else
                if current.size() < graph.vertices.size() then
                    for vertex in graph and not in current do
                        stack.push(current.append(vertex))
                    end for
                else
                    stack.push(current.append(depot))
                end if
            end if
        end if
    end while

    return best
end procedure
```

Analysis & Comments

- Runtime?
- A^* = admissible heuristic + priority queue (best first)
- best first vs depth first