Solving NP-complete problems

Petr Kurapov

Fall 2024

Agenda

- Exact solution (today)
  - Brute force search
  - Branch and bound
  - Dynamic programming, memoization
- Approximation
  - Greedy strategy
  - Heuristics, local search
  - Monte-Carlo
  - Meta-heuristics (genetic alg, annealing, ant colony, etc.)

Branch and Bound

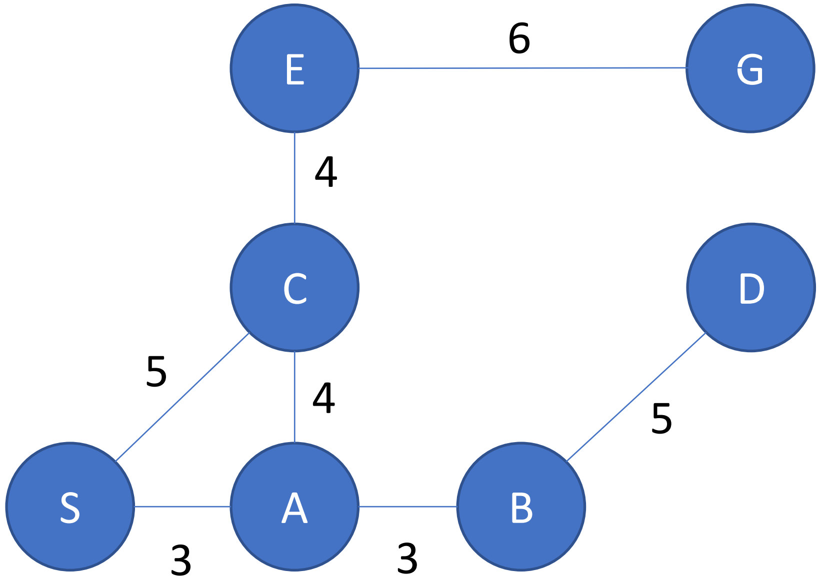
Idea:

- Branching: Assume a set G is split into subsets $G_i \leq G, i = 1 \dots r$: $\bigcup_r G_i = G$ - recursively = search tree
- Upper bound $UB_i \geq \min_{x \in G_i} f(x)$
- Lower bound $LB_i \leq \min_{x \in G_i} f(x)$

Construct several solutions sequentially and store a target function record. If G_i has lower bound $LB_i > Record$, then G_i doesn't contain optimal solution – do not explore the branch anymore.

Branch and Bound

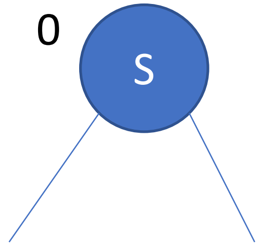
- Short path example for approach demonstration (S to D)



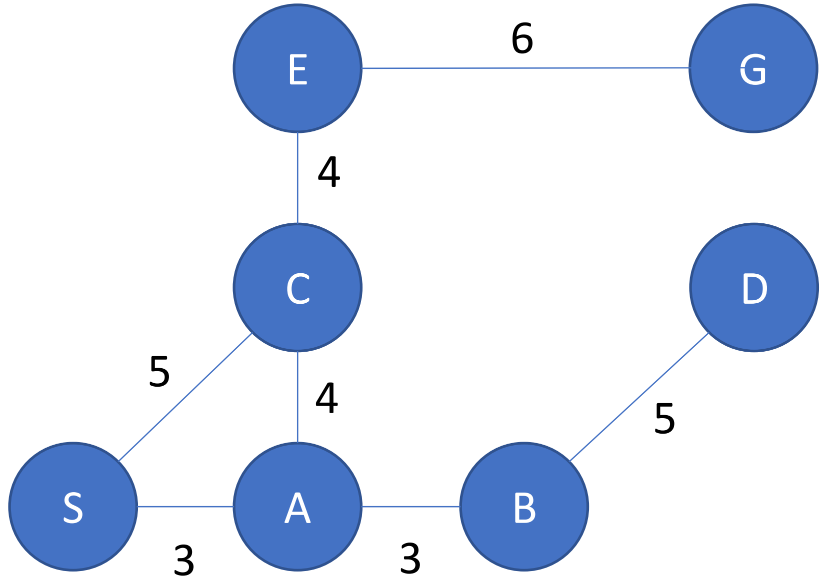
Example source: MIT 6.034

Branch and Bound

- Search tree

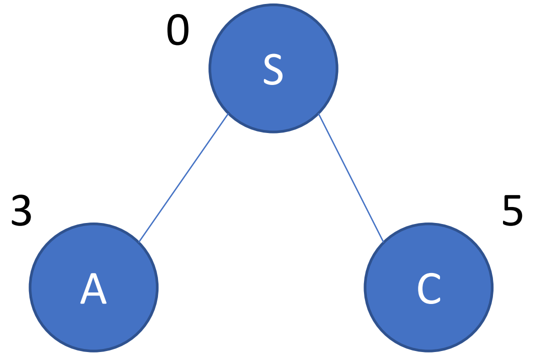


- Short path example for approach demonstration (S to D)



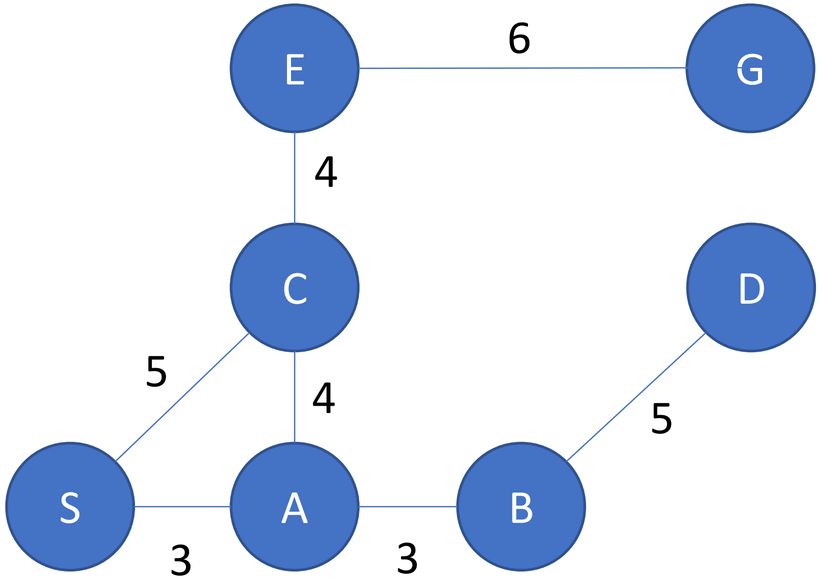
Branch and Bound

- Search tree



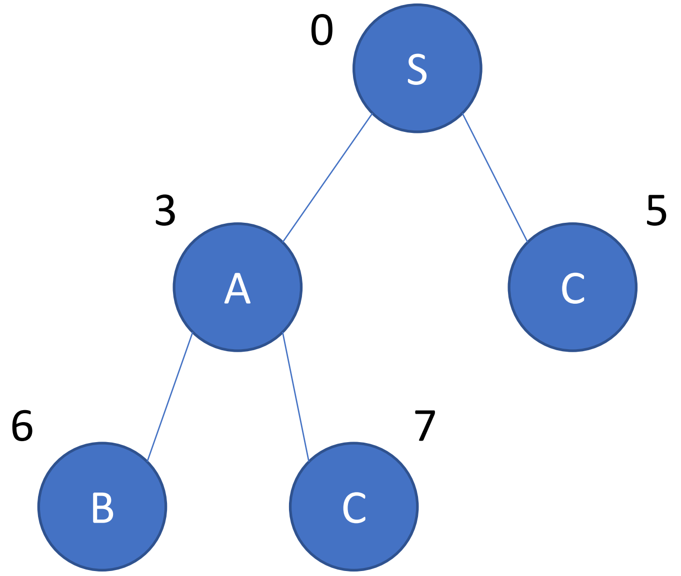
Sort nodes in lexical order

- Short path example for approach demonstration (S to D)

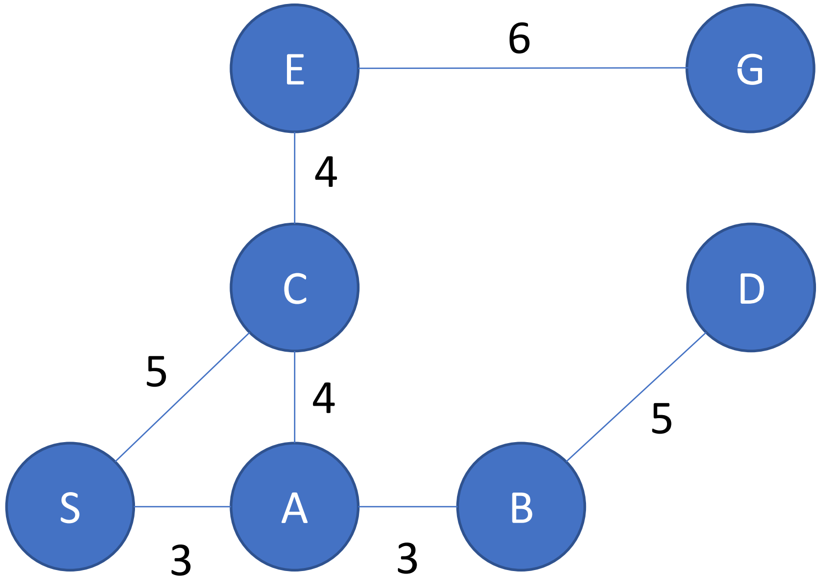


Branch and Bound

- Search tree

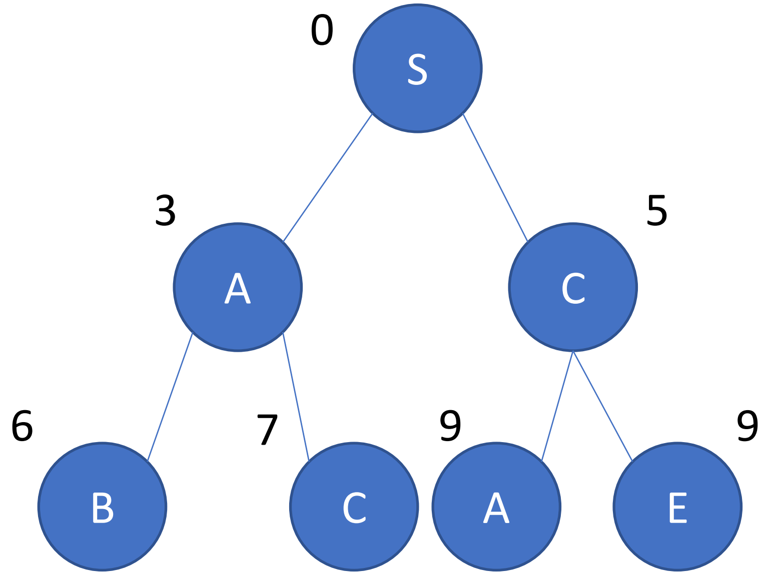


- Short path example for approach demonstration (S to D)

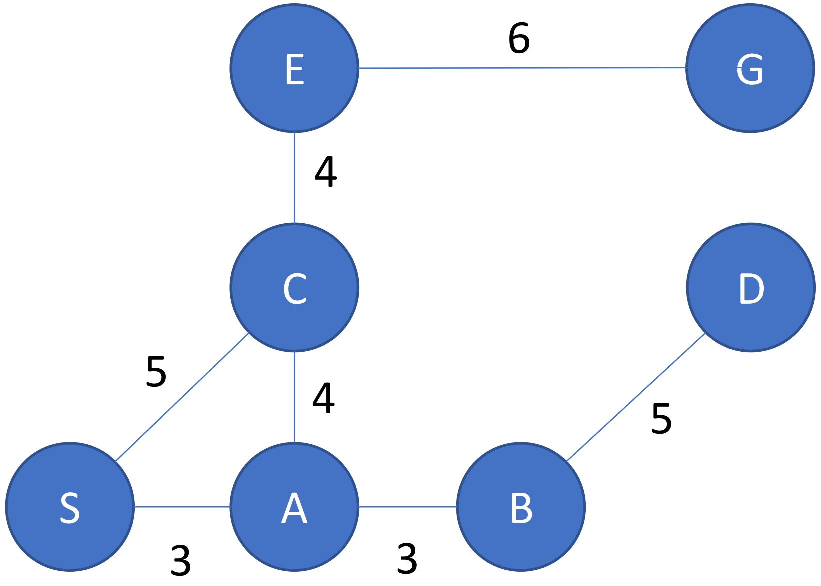


Branch and Bound

- Search tree

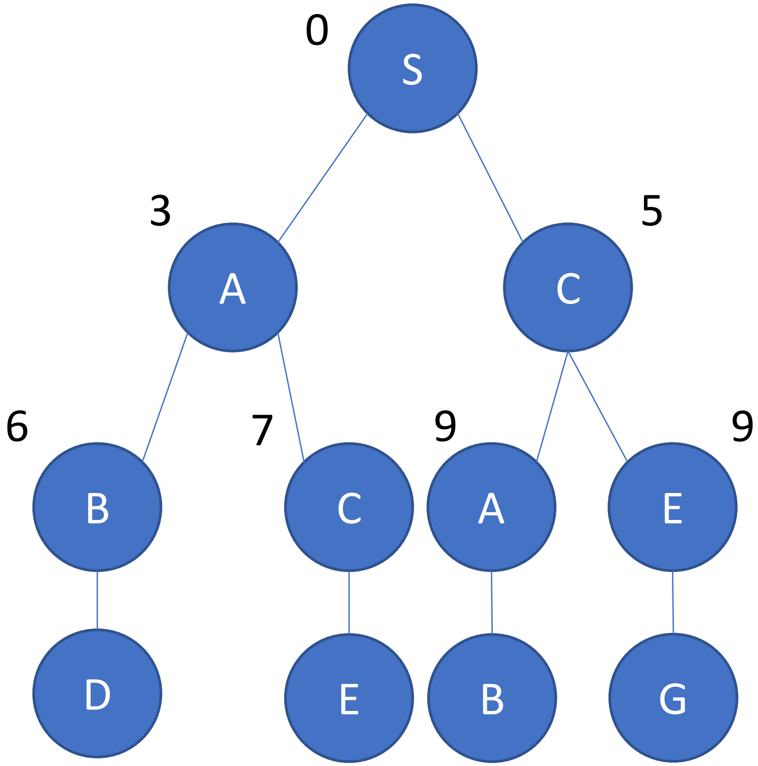


- Short path example for approach demonstration (S to D)



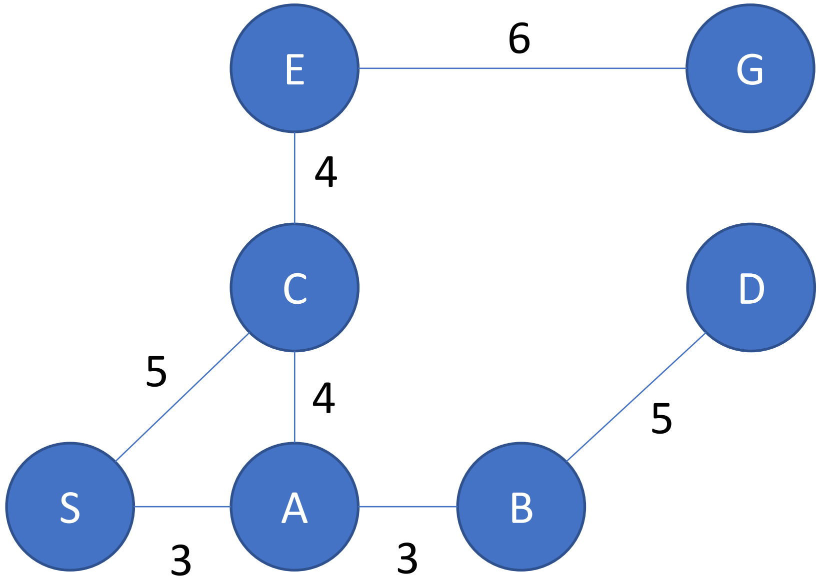
Branch and Bound

- Search tree



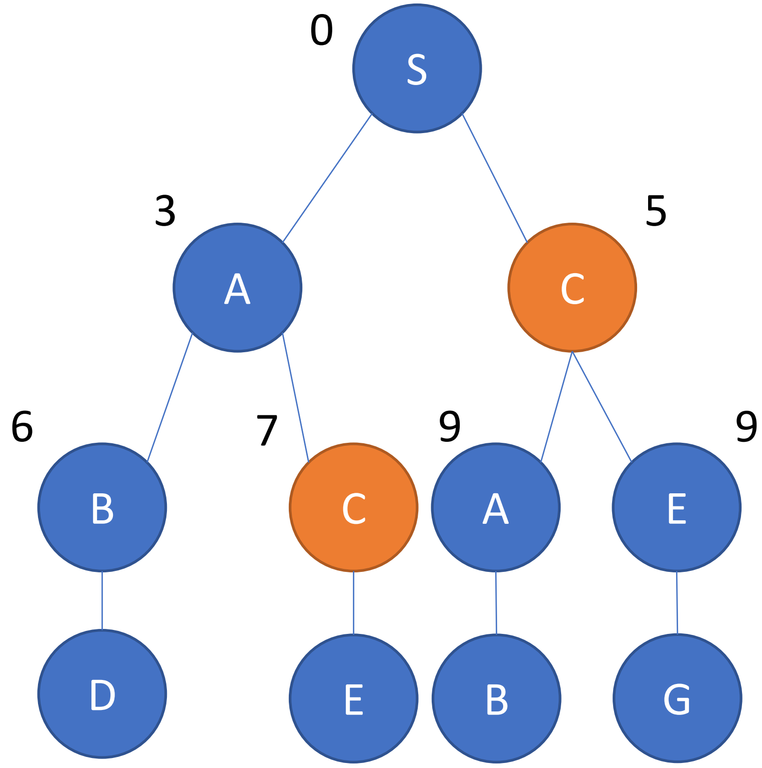
- Complete search tree
- Back tracking

- Short path example for approach demonstration (S to D)



Branch and Bound

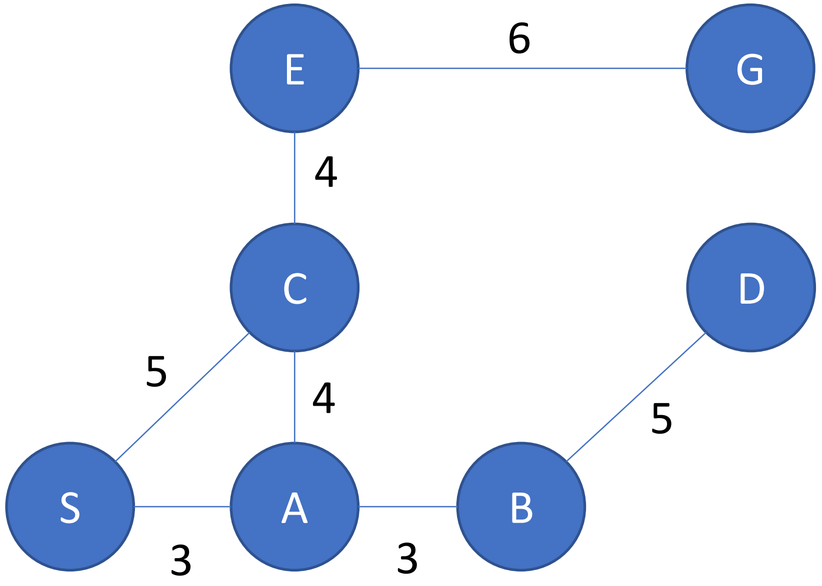
- Search tree



- Complete search tree
- Back tracking

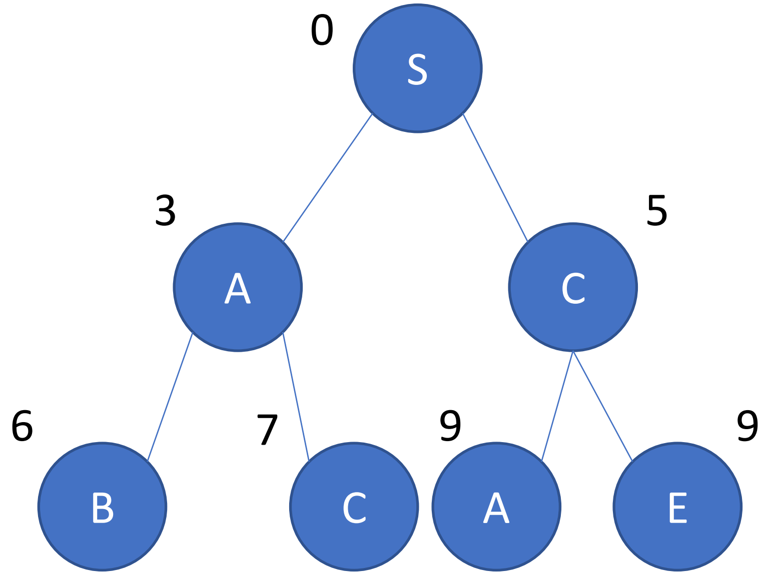
Extremely stupid:
- Visiting the same nodes
- Can't tell if the solution gets better or not

- Short path example for approach demonstration (S to D)

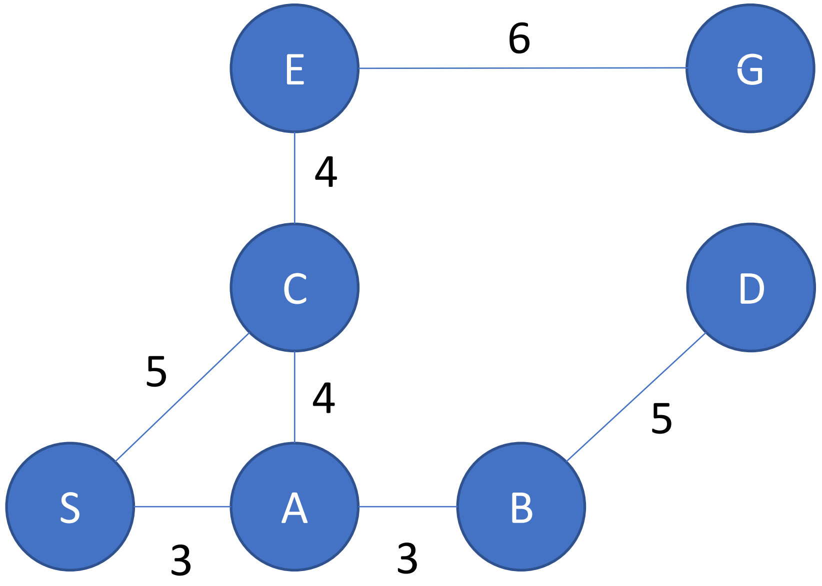


Branch and Bound

- Search tree

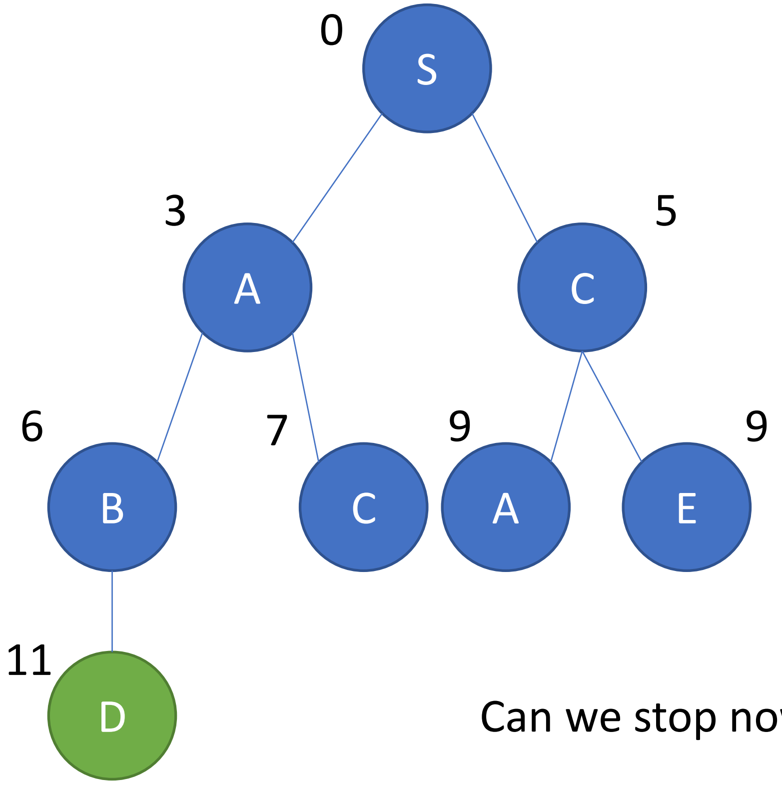


- Short path example for approach demonstration (S to D)



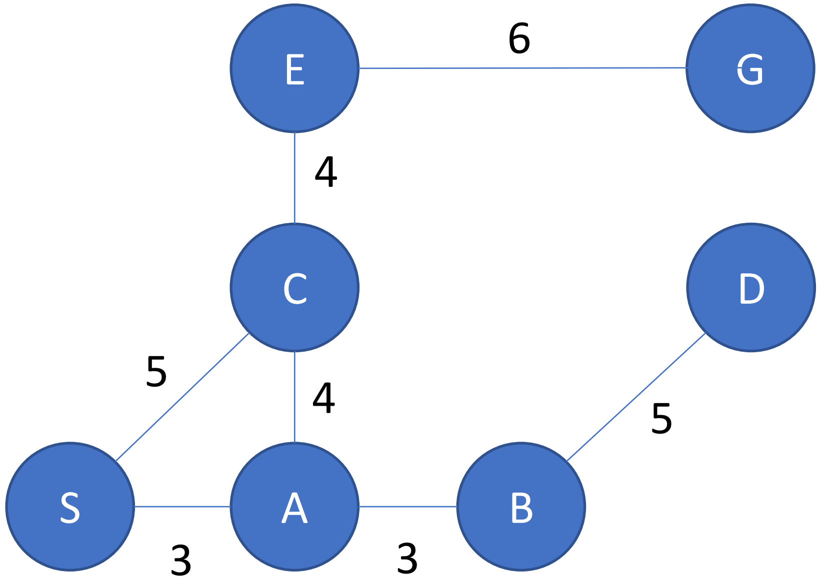
Branch and Bound

- Search tree



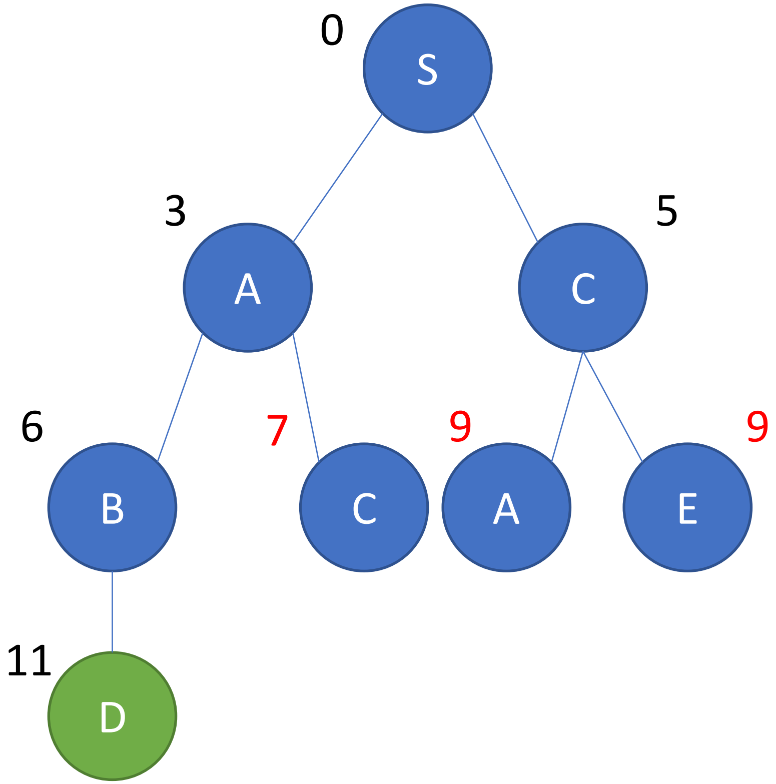
Can we stop now?

- Short path example for approach demonstration (S to D)



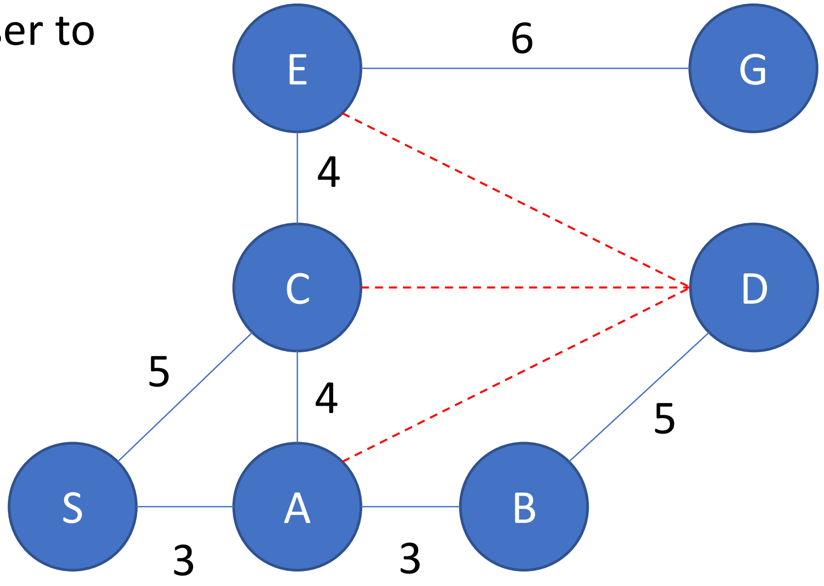
Branch and Bound

- Search tree



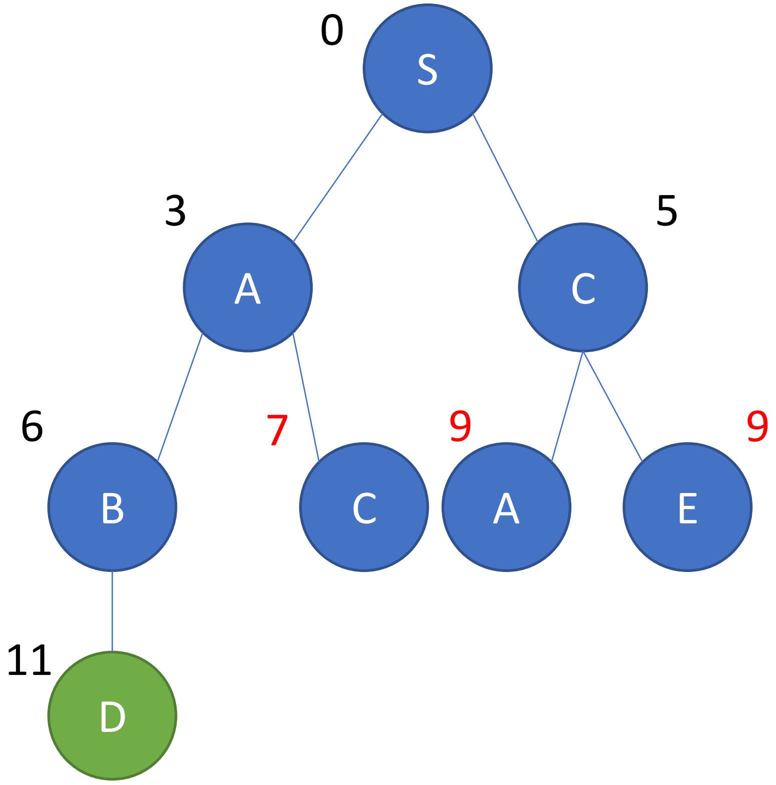
- Short path example for approach demonstration (S to D)

Heuristic: being closer to the goal is better?



Branch and Bound

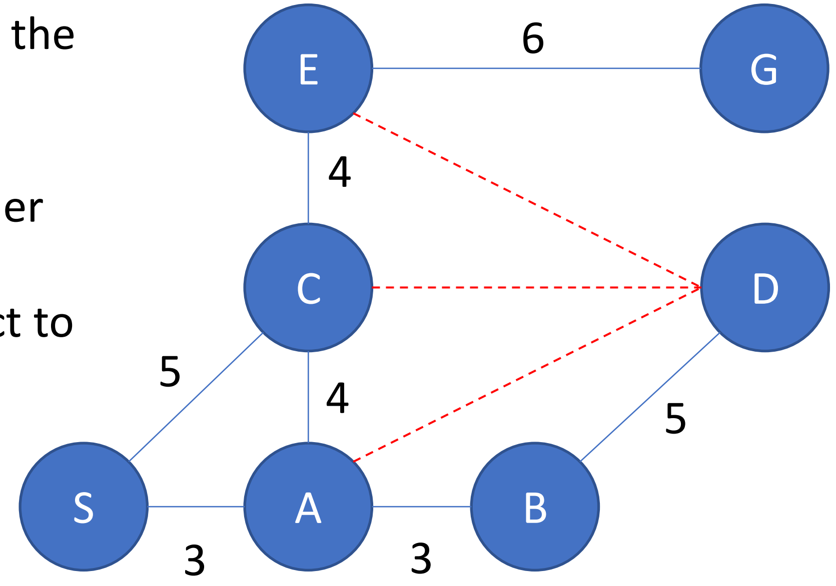
- Search tree



- Short path example for approach demonstration (S to D)

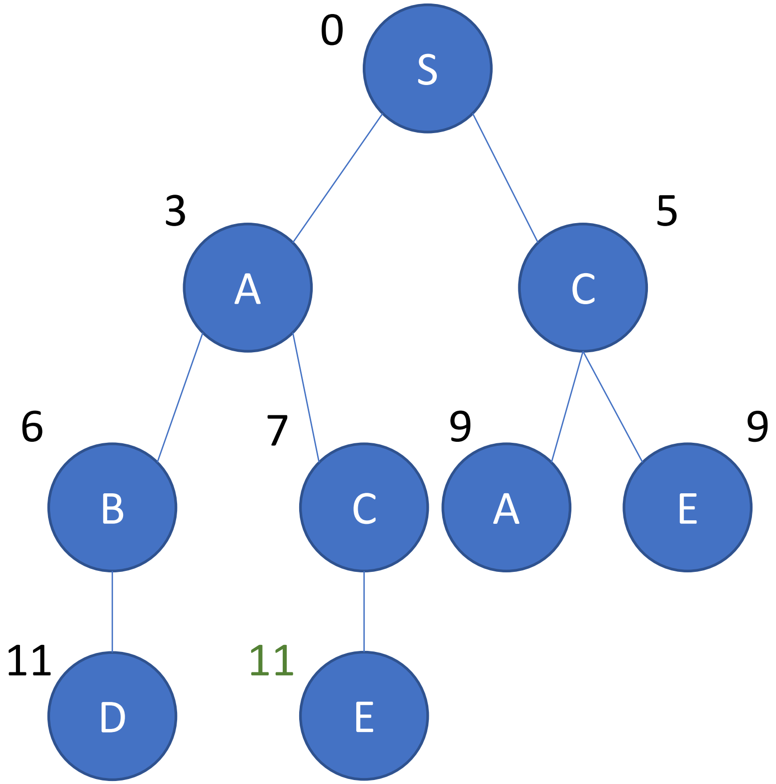
Heuristic: being closer to the goal is better?

DFS: heuristic guided order instead of lexical
BFS: beam search (restrict to beam width)

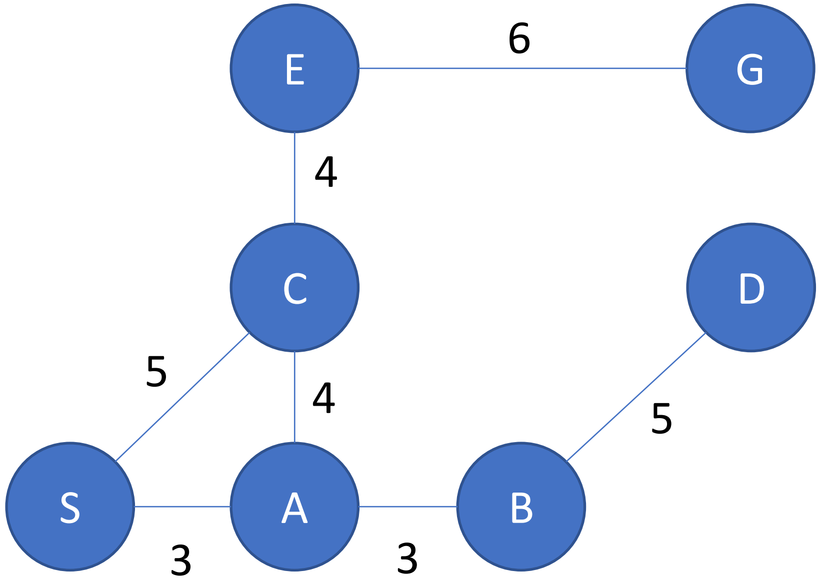


Branch and Bound

- Search tree

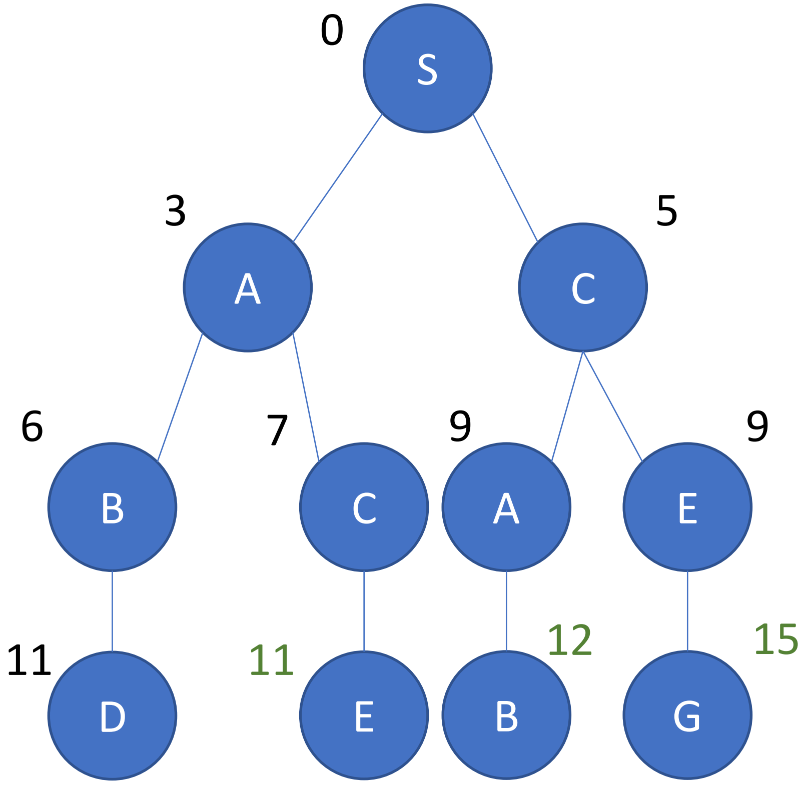


- Short path example for approach demonstration (S to D)

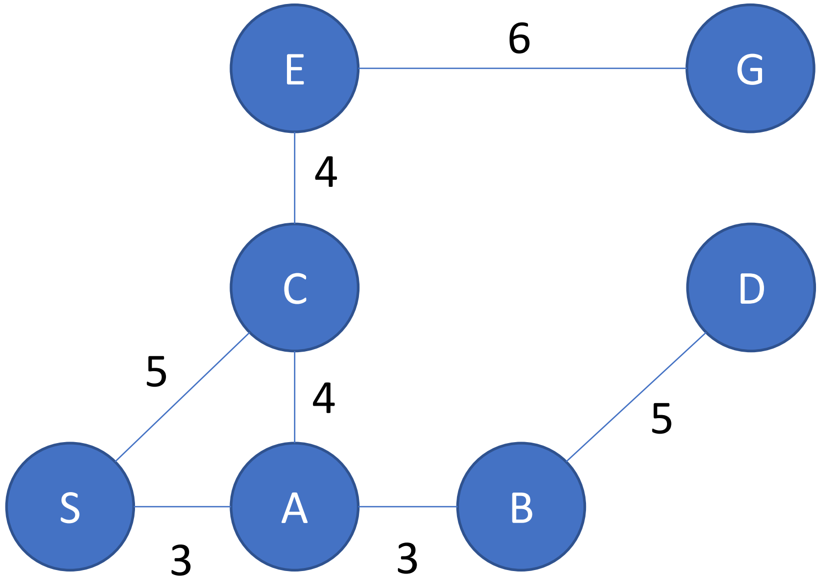


Branch and Bound

- Search tree

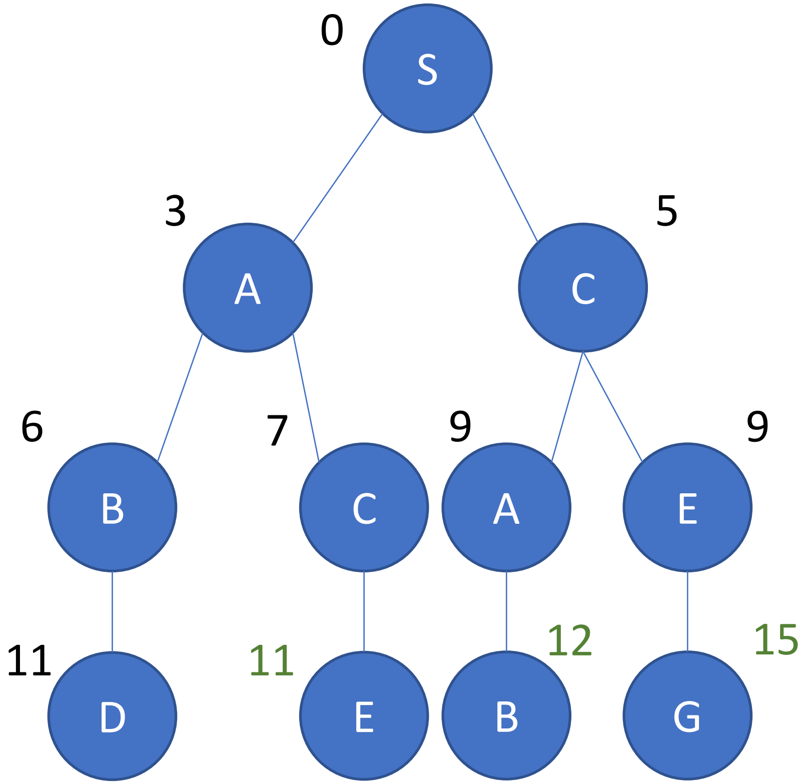


- Short path example for approach demonstration (S to D)



Branch and Bound

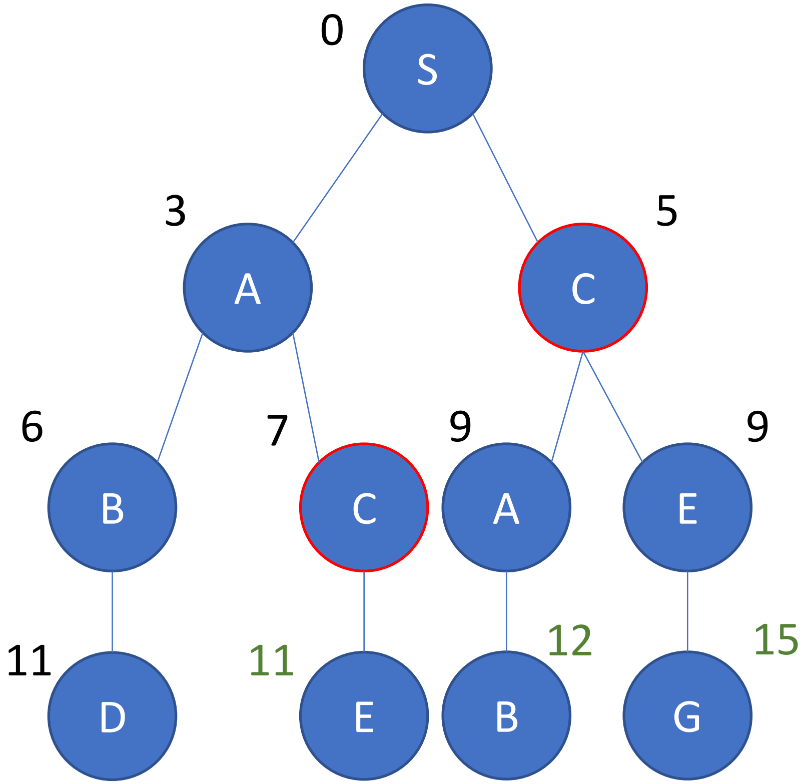
- Search tree



- Create a queue
  - LIFO
  - FIFO
  - Cost based
- Test a path
  - Done if found*
- Extend the path & sort* new queue items

Branch and Bound

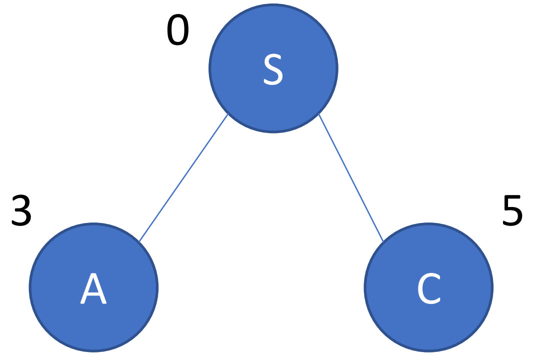
- Search tree



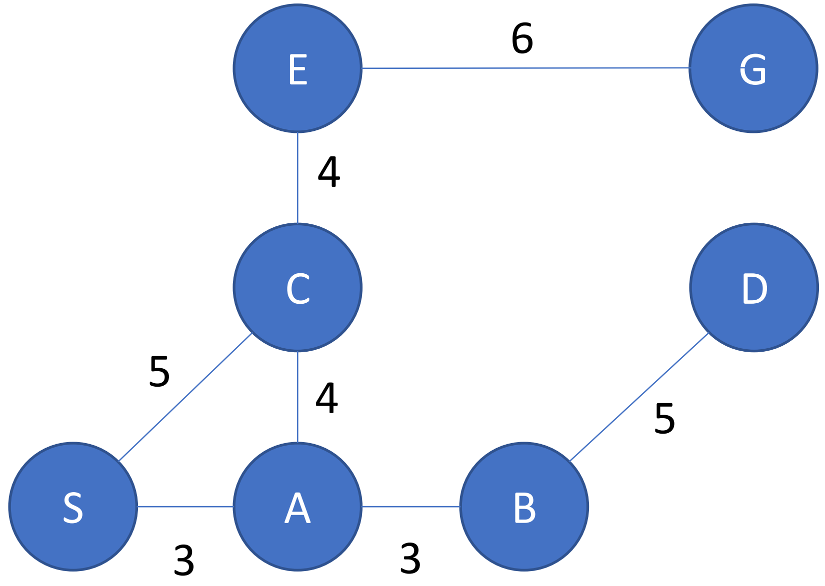
- Create a queue
  - LIFO
  - FIFO
  - Cost based
- Test a path
  - Done if found*
- Extend the path & sort* new queue items
  - Don’t extend nodes that have been extended

Branch and Bound

- Search tree

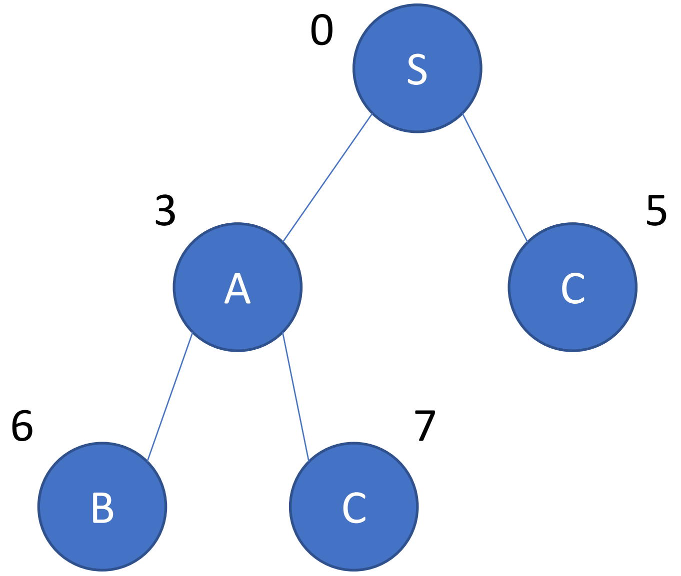


- Short path example for approach demonstration (S to D)

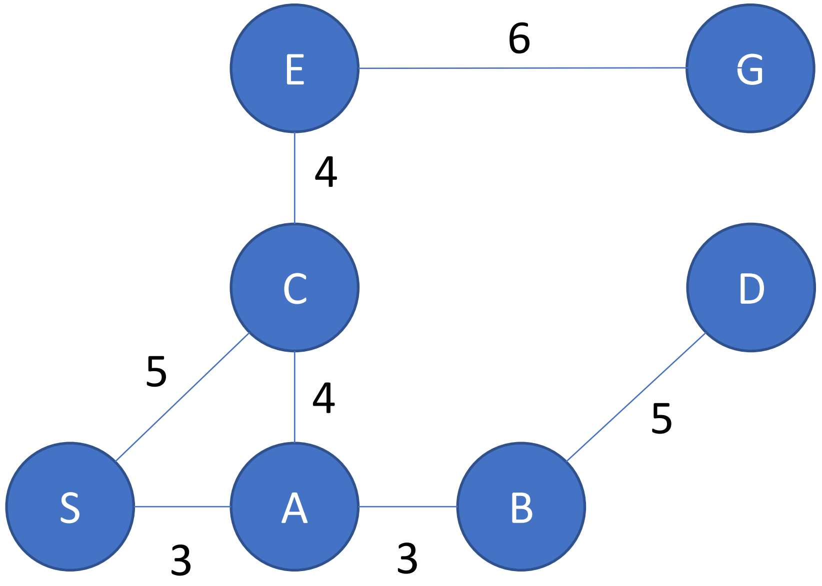


Branch and Bound

- Search tree

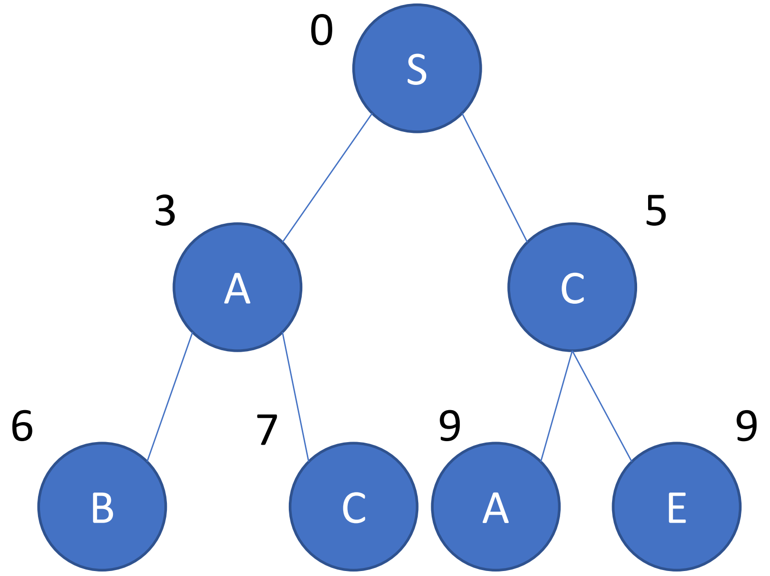


- Short path example for approach demonstration (S to D)

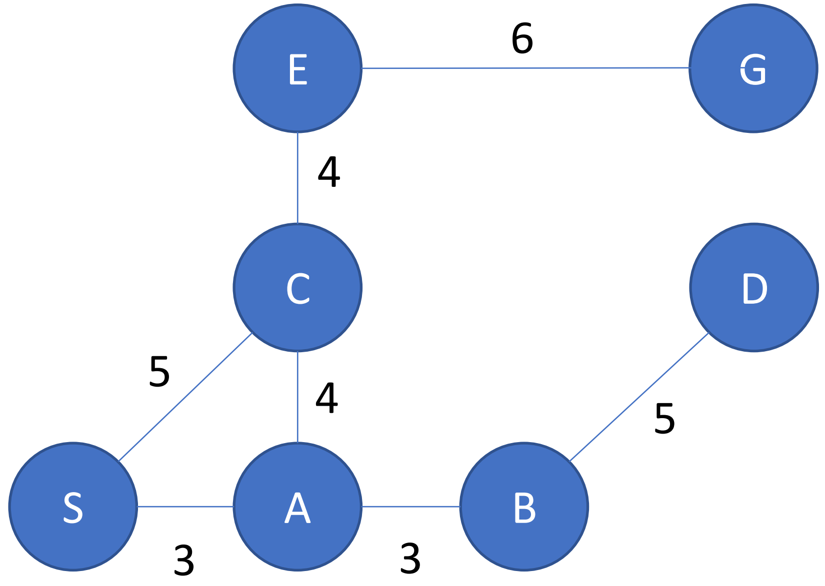


Branch and Bound

- Search tree

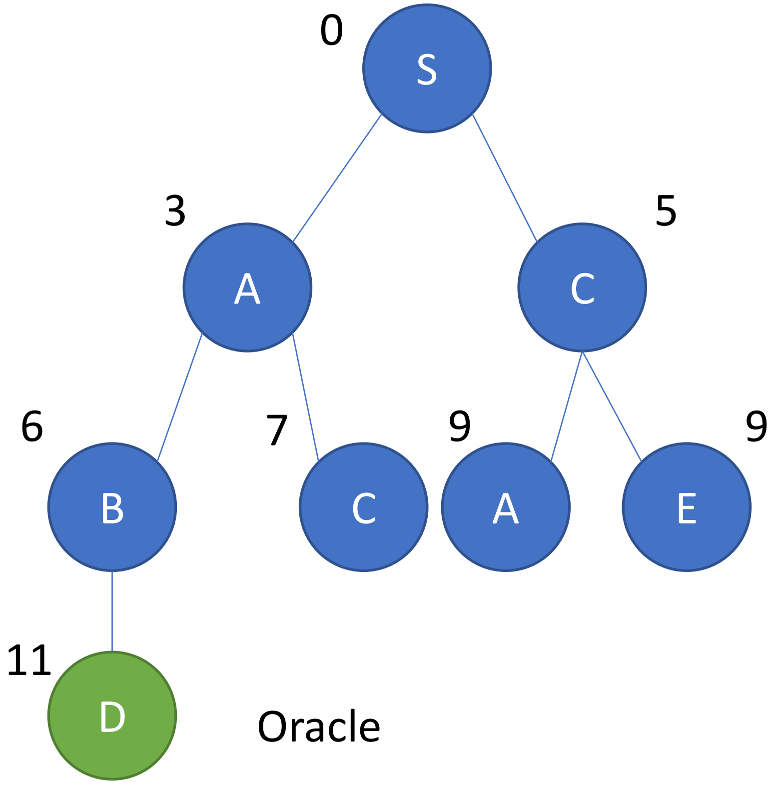


- Short path example for approach demonstration (S to D)

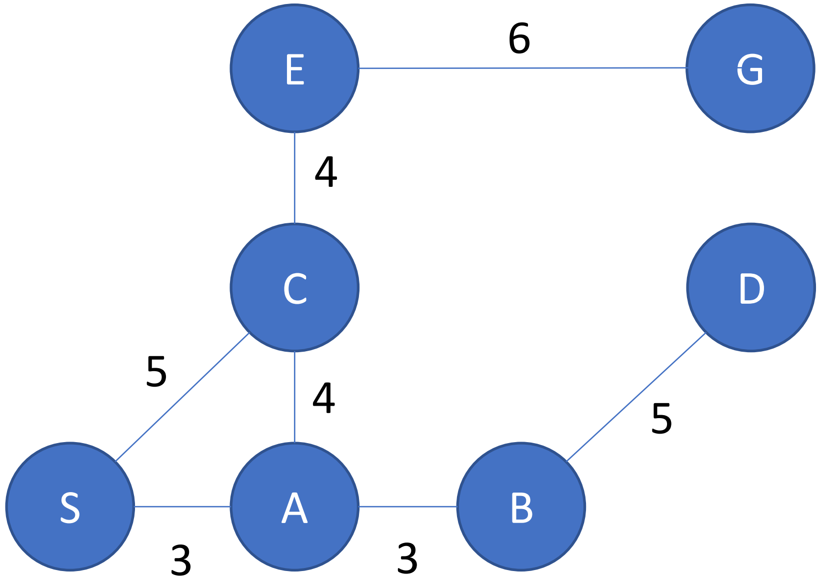


Branch and Bound

- Search tree

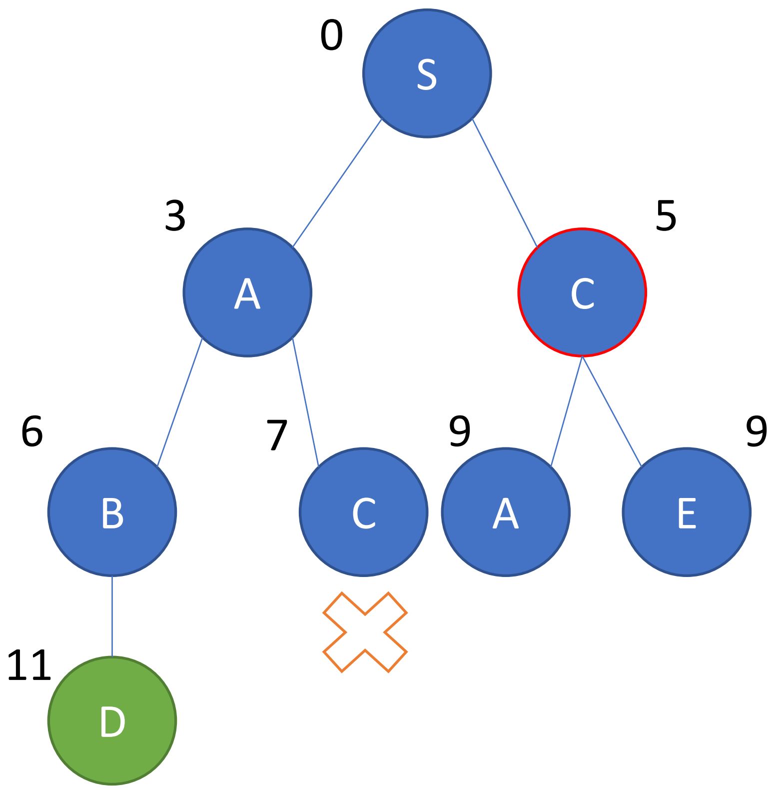


- Short path example for approach demonstration (S to D)

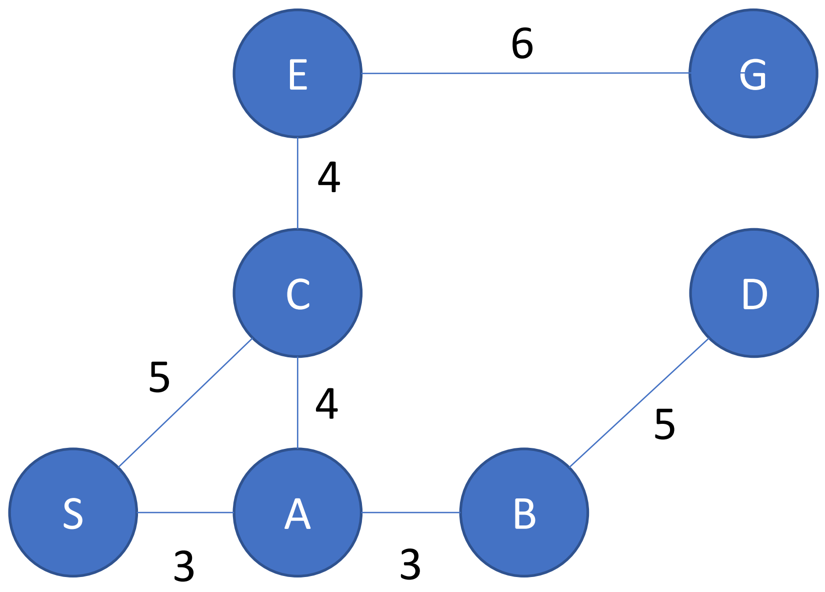


Branch and Bound

- Search tree

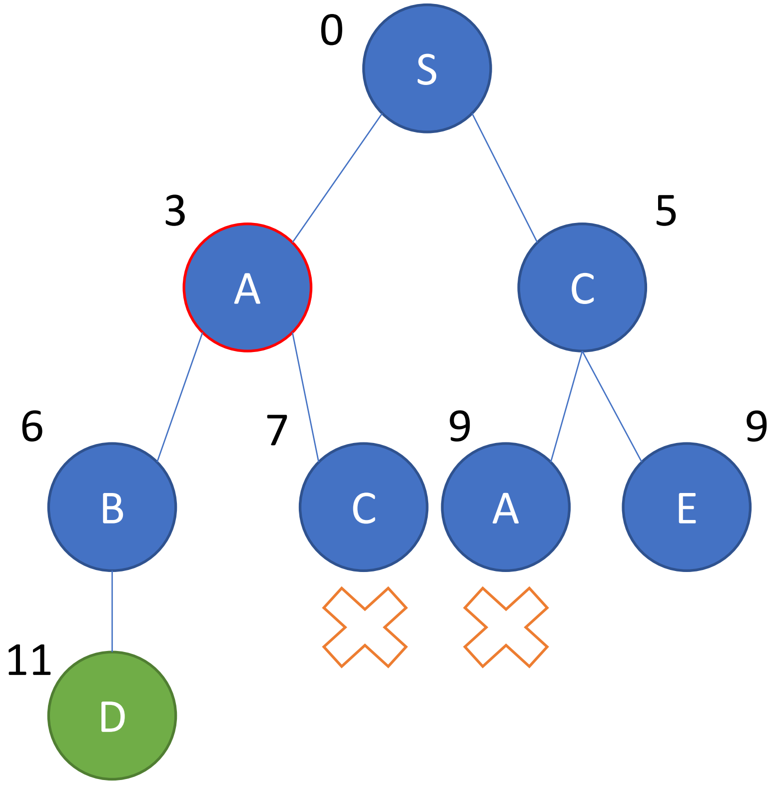


- Short path example for approach demonstration (S to D)

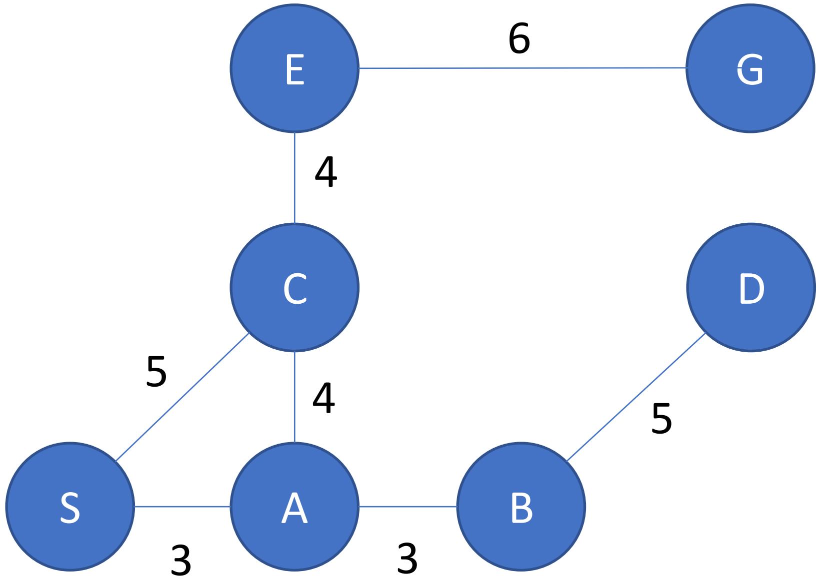


Branch and Bound

- Search tree

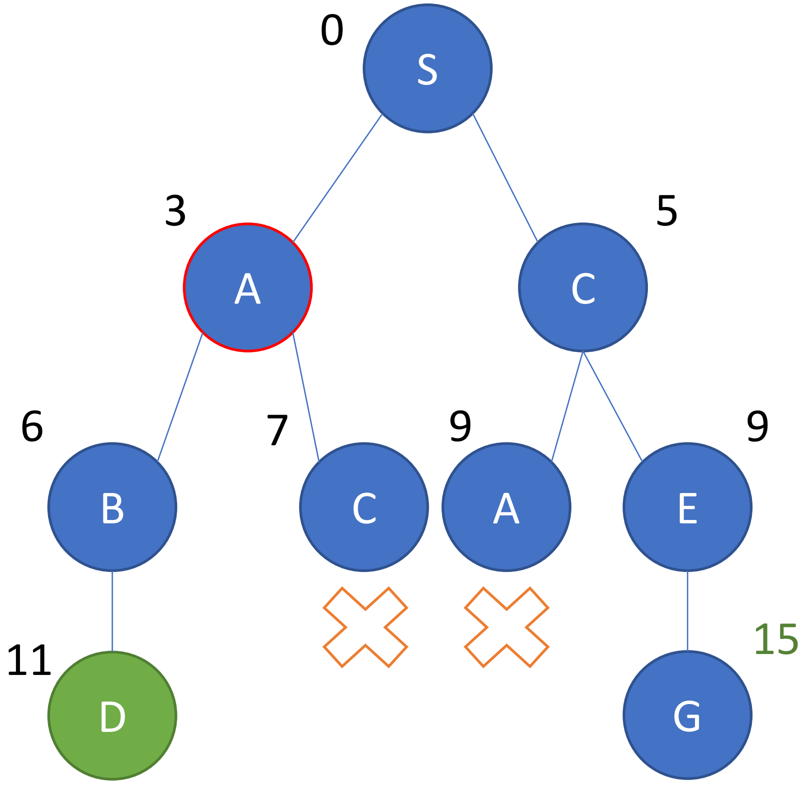


- Short path example for approach demonstration (S to D)

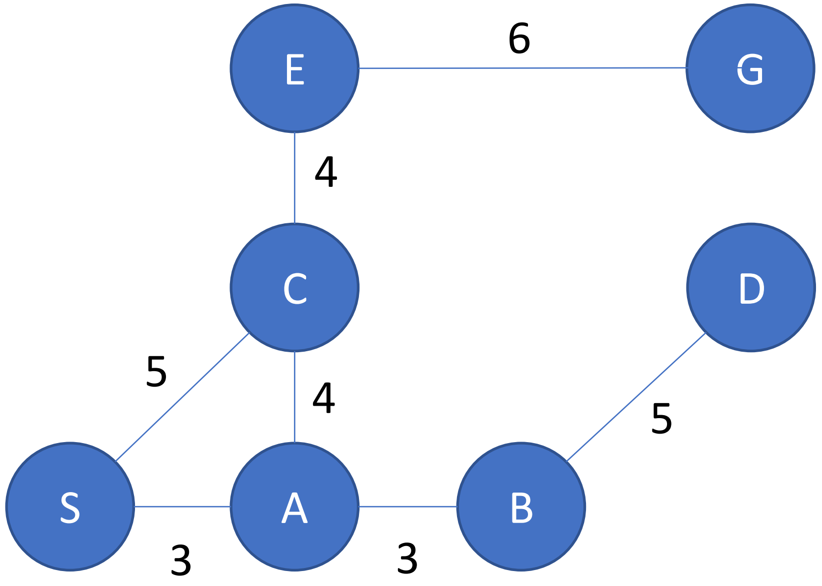


Branch and Bound

- Search tree



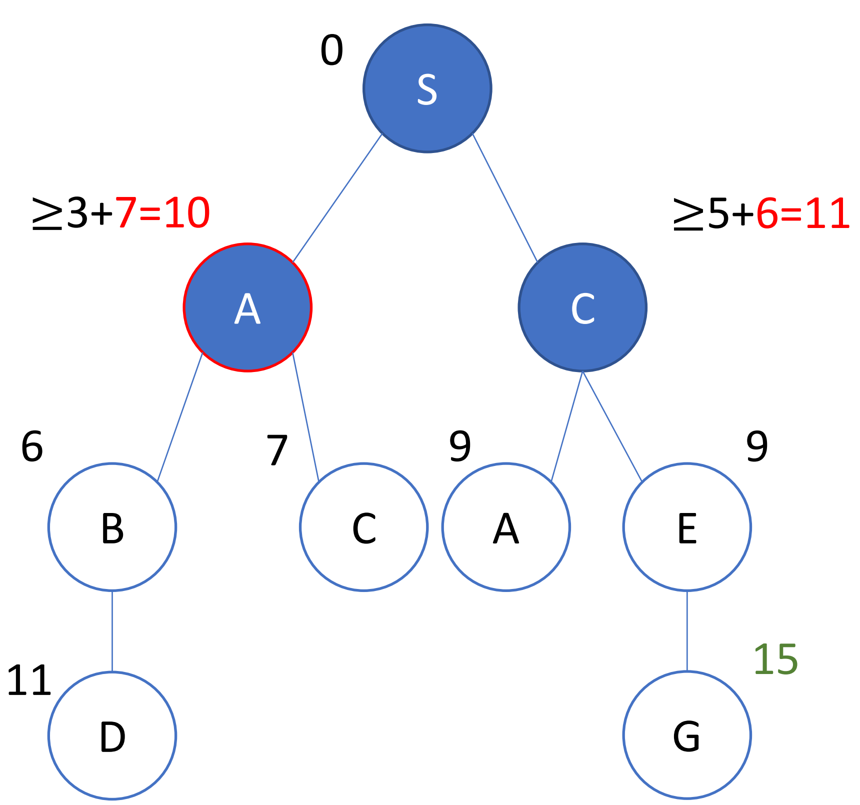
- Short path example for approach demonstration (S to D)



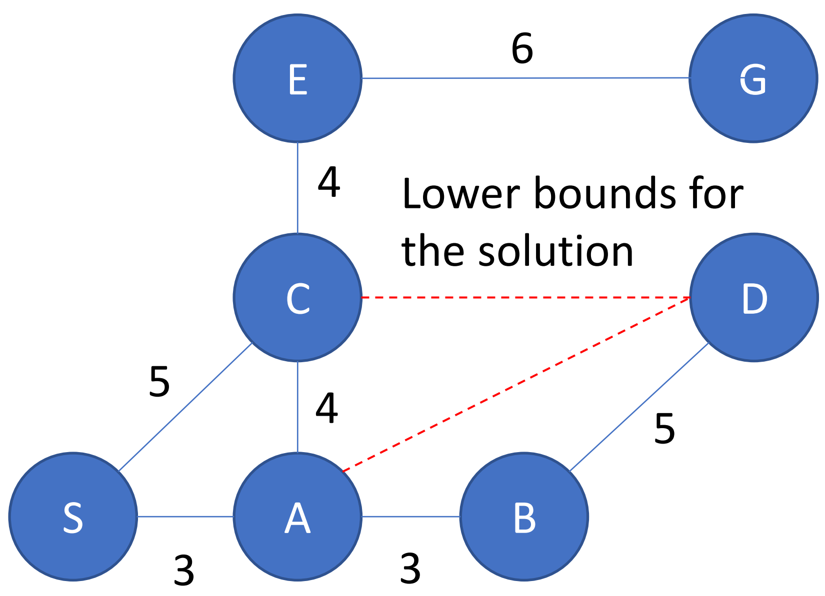
Dead horse principle

Branch and Bound

- Search tree



- Short path example for approach demonstration (S to D)



Dynamic programming

- Divide and concur vs DP
- Optimization problems

- Optimal solution structure description
- Initialization values
- Transition function
- Order

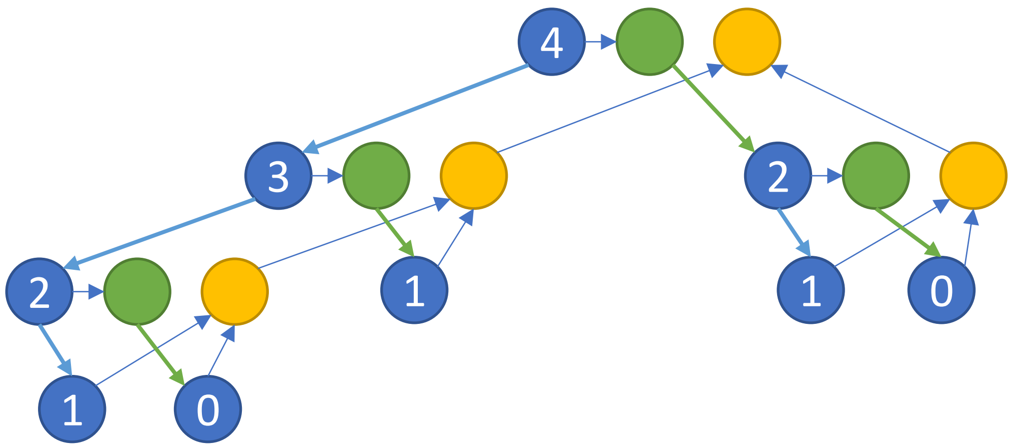
Idea: given a problem we don’t know how to solve – split it into subproblems (we don’t know how to solve either)

Dynamic programming

Fibonacci example:

```
def fib(n):
    if (n <= 2):
        return n
    return fib(n-1)+fib(n-2)
```

- Fib(4)

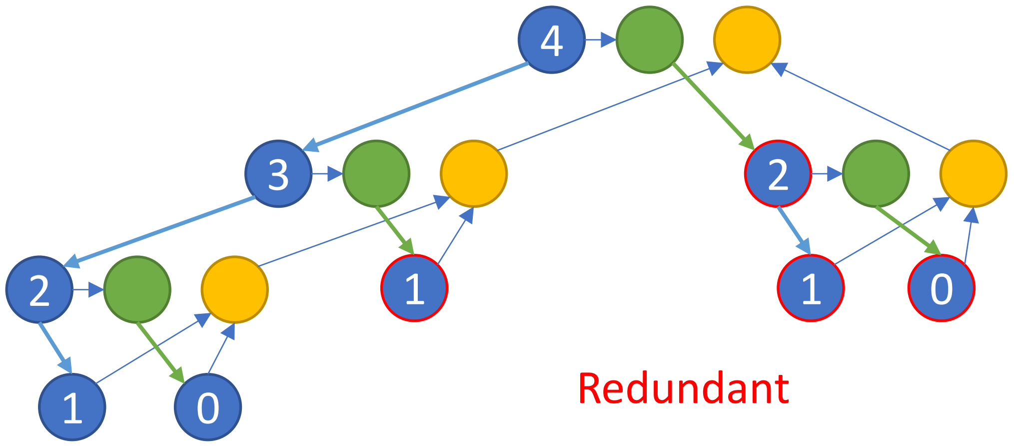


Dynamic programming

Fibonacci example:

```
def fib(n):
    if (n <= 2):
        return n
    return fib(n-1)+fib(n-2)
```

- Fib(4)

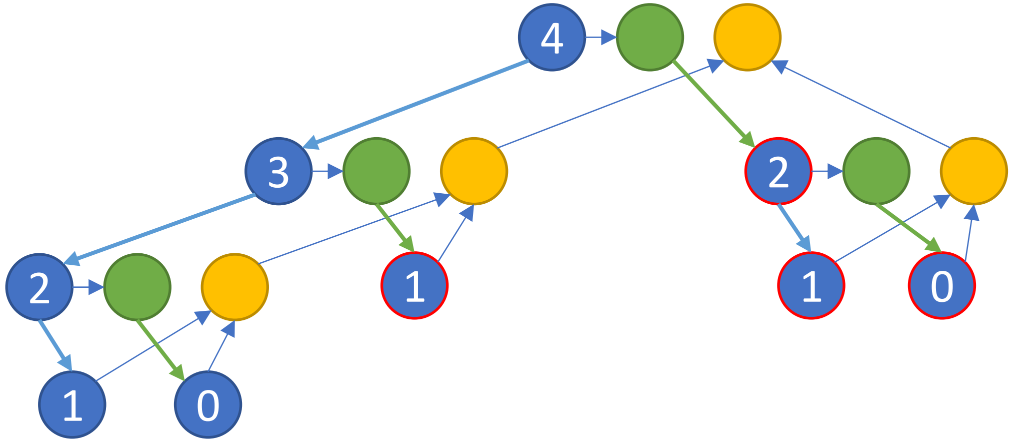


Dynamic programming

Fibonacci example:

```
def fib(n):
    if (n <= 2):
        return n
    return fib(n-1)+fib(n-2)
```

- Fib(4)



Dynamic programming

- Forward order
- Backward order
- Lazy dynamic

Dynamic programming

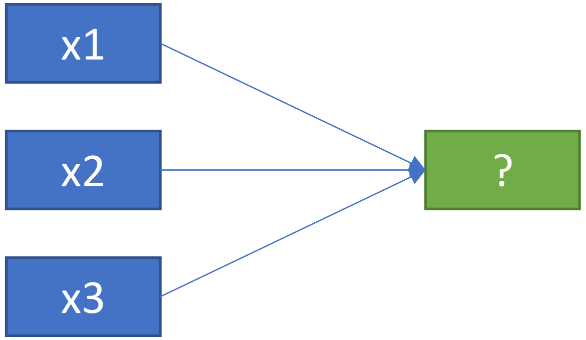
- **Forward order**
- Backward order
- Lazy dynamic

```
fib[0] = fib[1] = 1
for (2..n):
  fib[i] = fib[i-1]+fib[i-2]
```

Dynamic programming

- **Forward order**
- Backward order
- Lazy dynamic

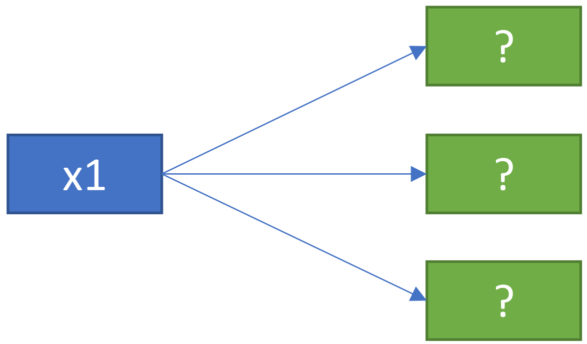
```
fib[0] = fib[1] = 1
for (2..n):
  fib[i] = fib[i-1]+fib[i-2]
```



Dynamic programming

- Forward order
- **Backward order**
- Lazy dynamic

```
fib[0] = 1
for (0..n):
    fib[i+1] += fib[i]
    fib[i+2] += fib[i]
```



Dynamic programming

- Forward order
- Backward order
- **Lazy dynamic**

```
def fib(i):
    if i <= 2: return 1
    if fib[i] == -1:
        fib[i] =
            fib(i-1)+fib(i-2)
    return fib[i]
```

Example: numbers

Problem:

- Given positive N, calculate the number of permutations of length N that do not contain consequent ones (1 immediately followed by 1)

- 010010100
- 001101011

For N = 2: 00, 01, 10, 11 – the number of legal permutations is 3

Example: numbers

Problem:

- Given positive N, calculate the number of permutations of length N that do not contain consequent ones (1 immediately followed by 1)

$$\begin{cases} x_1 x_2 \ldots x_{n-1} 0, -2\ options\ \{1,0\} \\ x_1 x_2 \ldots x_{n-1} 1, -\quad 1\ option\ \{0\} \end{cases}$$

Example: numbers

Problem:

- Given positive N, calculate the number of permutations of length N that do not contain consequent ones (1 immediately followed by 1)

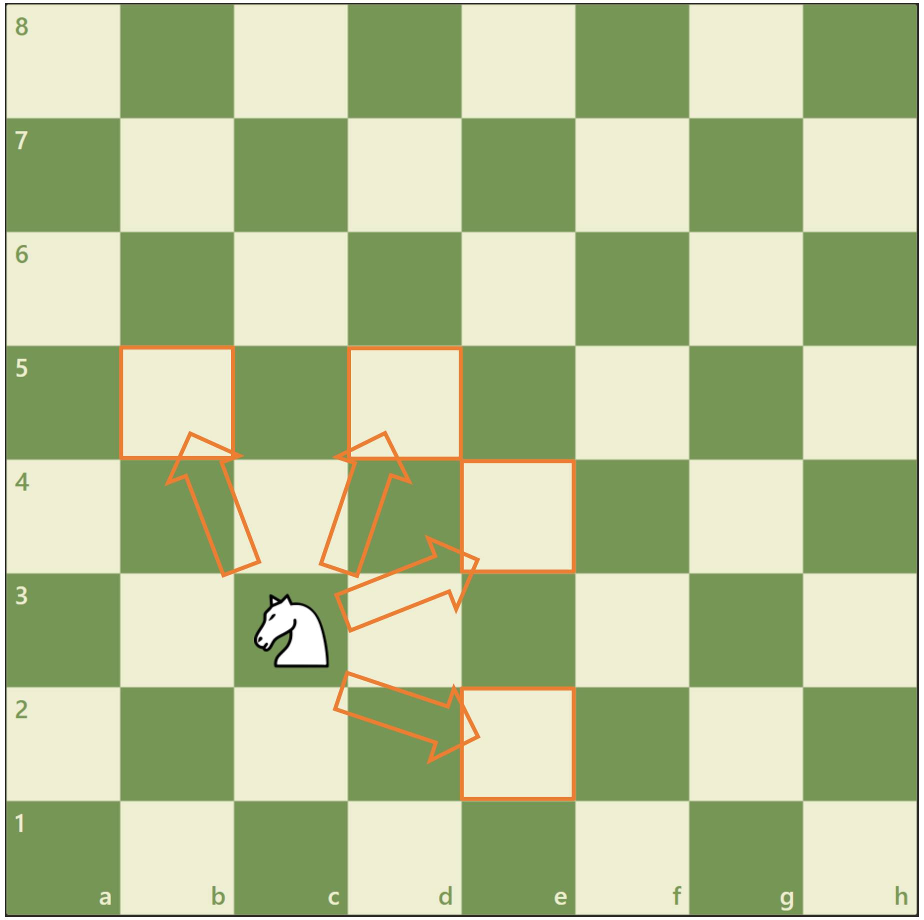
$$\begin{cases} x_1 x_2 \dots x_{n-1} 0, -2\ options\ \{1,0\} \\ x_1 x_2 \dots x_{n-1} 1, - \quad 1\ option\ \{0\} \end{cases}$$

$$f(n) = f(n-1) + f(n-2)$$

Example: knight

Problem:

- How many possible ways to get from a1 to h8 when the knight moves as shown?

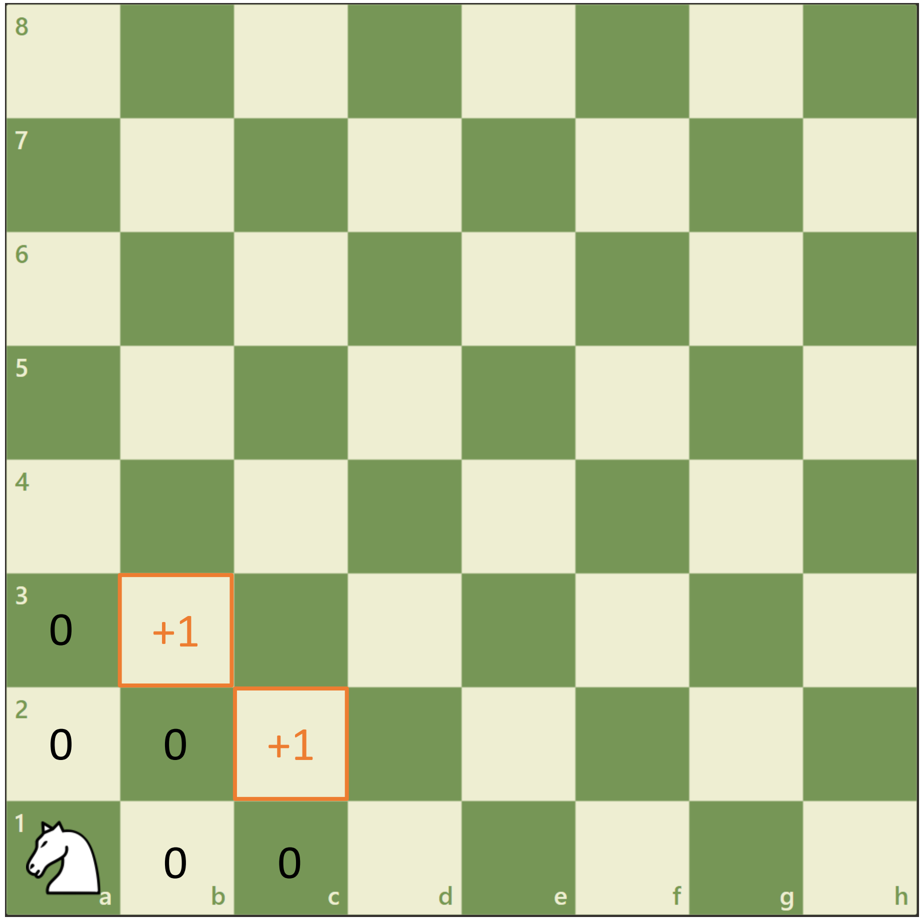


Board source: https://www.chess.com/

Example: knight

Problem:

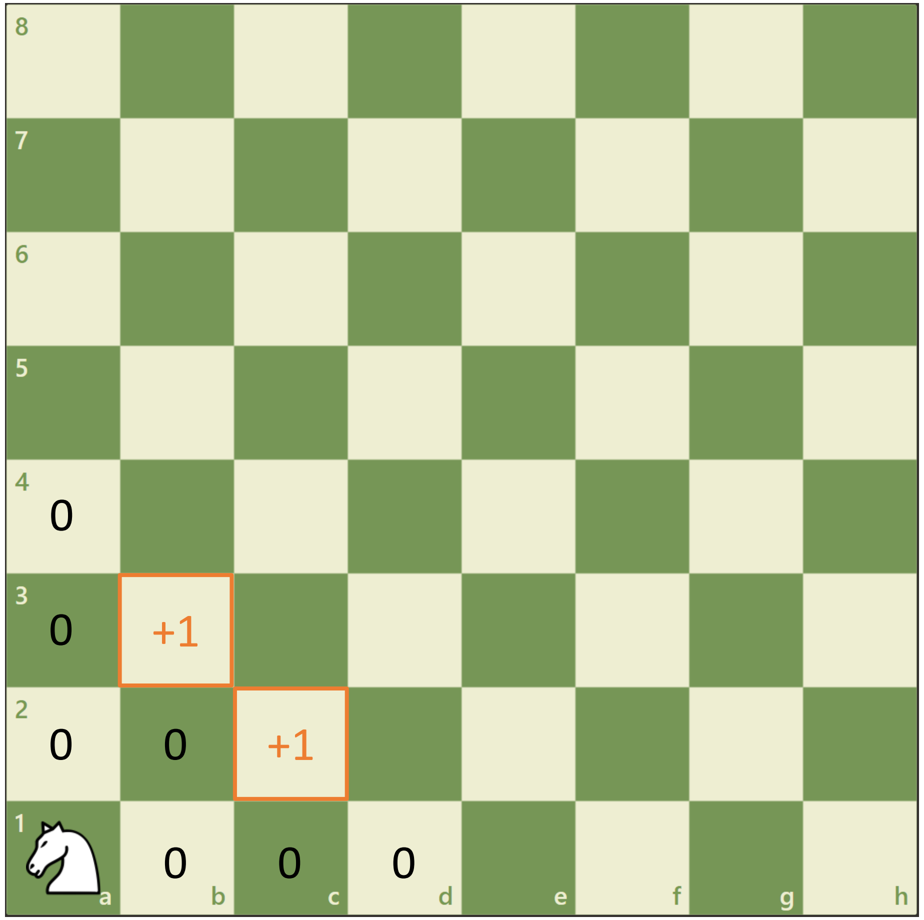
- How many possible ways to get from a1 to h8 when the knight moves as shown?
- Backward order: take current position and update dependents



Example: knight

Problem:

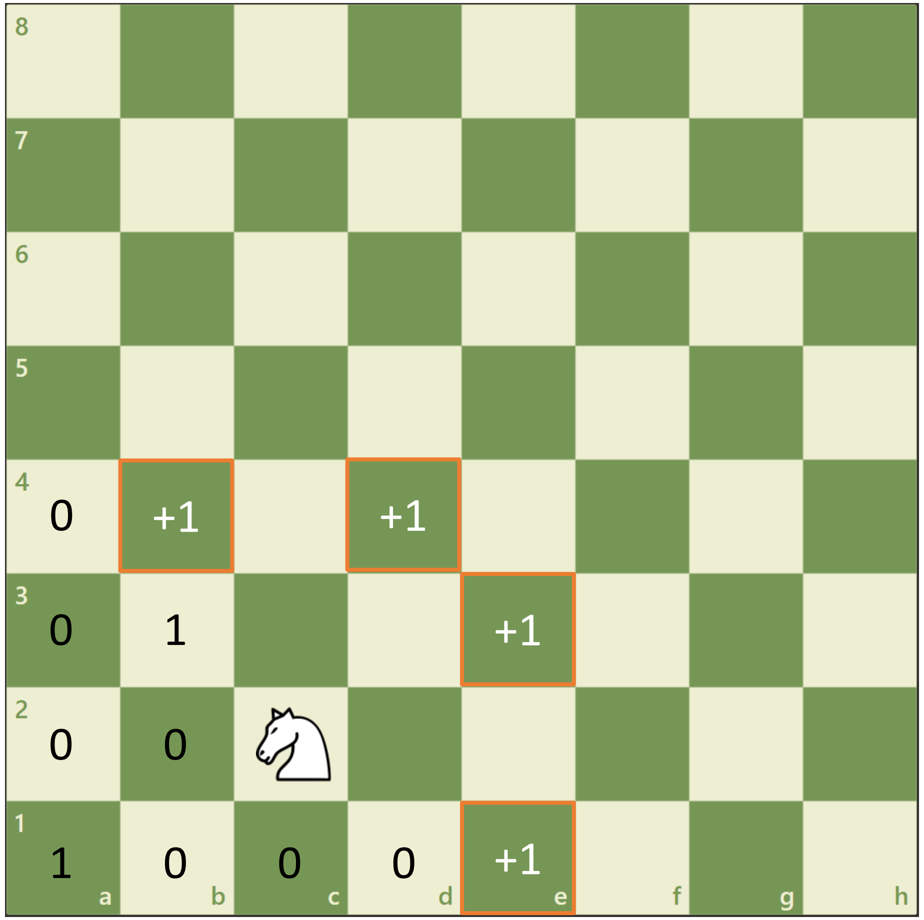
- How many possible ways to get from a1 to h8 when the knight moves as shown?
- Backward order: take current position and update dependents



Example: knight

Problem:

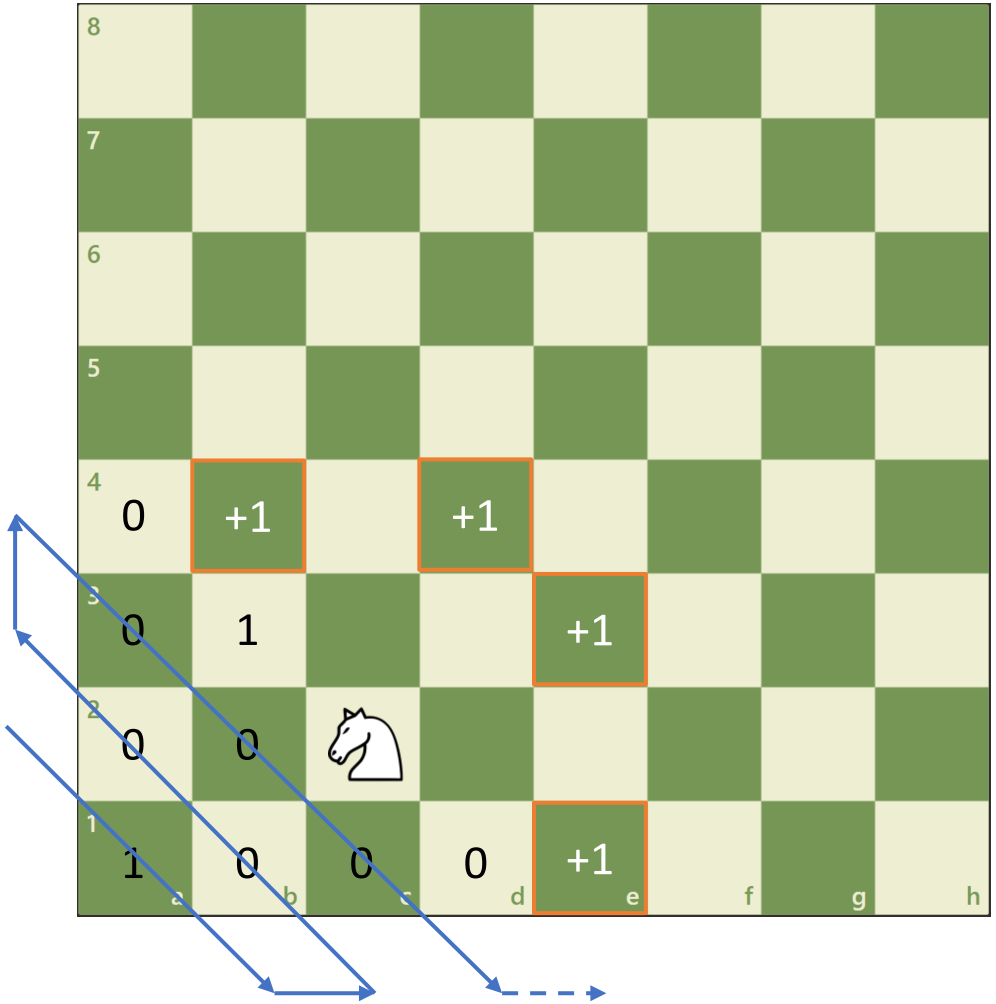
- How many possible ways to get from a1 to h8 when the knight moves as shown?
- Backward order: take current position and update dependents



Example: knight

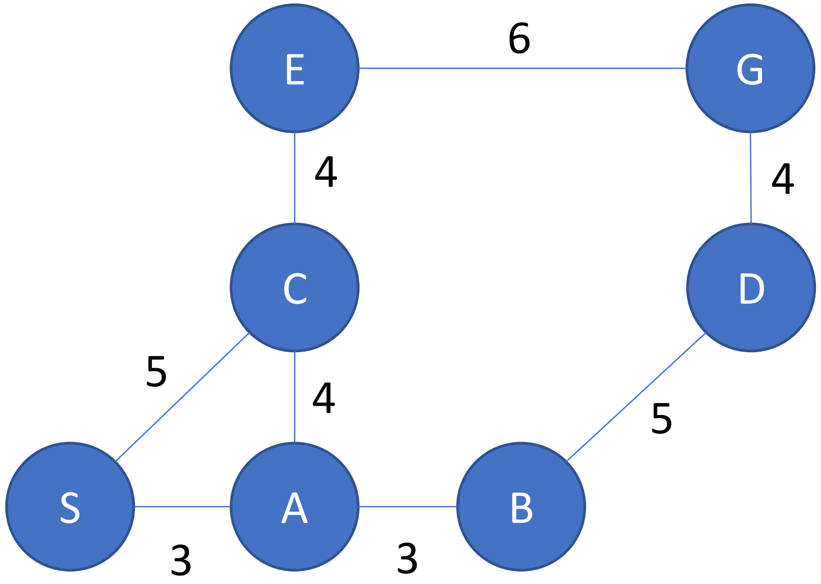
Problem:

- How many possible ways to get from a1 to h8 when the knight moves as shown?
- Backward order: take current position and update dependents
- Traverse order is important



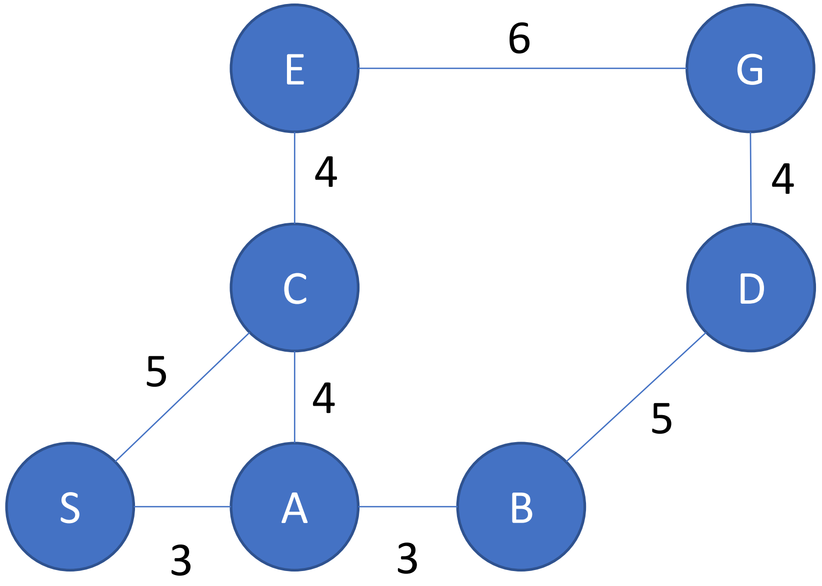
Example: Hamiltonian path (TSP)

- Hamiltonian path – graph that visits each vertex exactly once (example from [3])



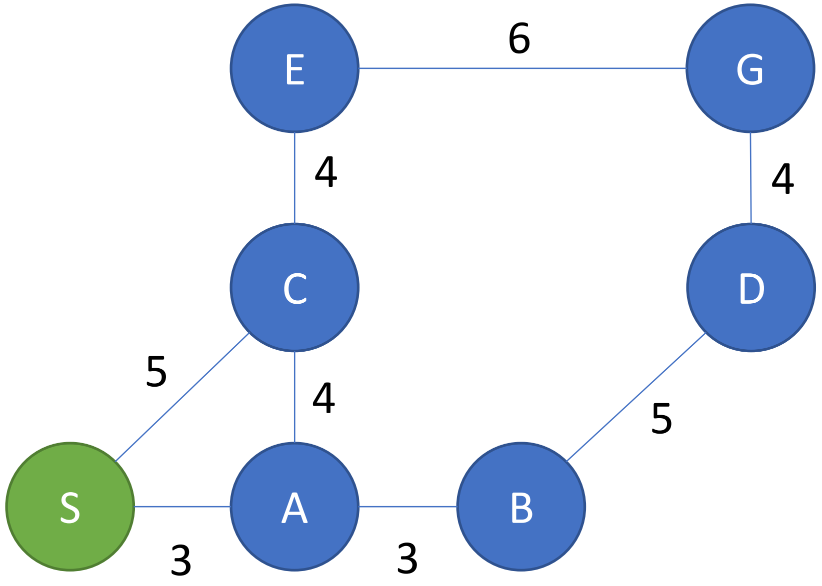
Example: Hamiltonian path (TSP)

- Hamiltonian path – graph that visits each vertex exactly once
- Idea: solve it for subgraphs



Example: Hamiltonian path (TSP)

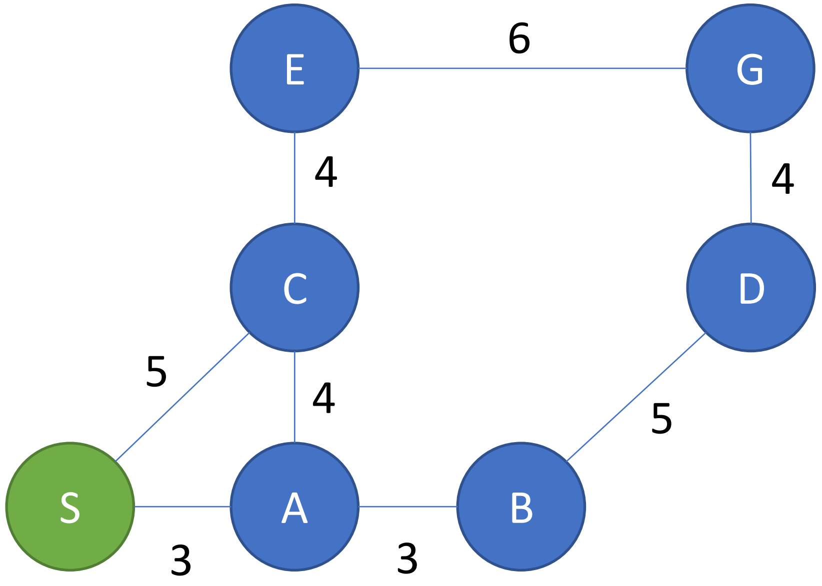
- Hamiltonian path – graph that visits each vertex exactly once
- Idea: solve it for subgraphs
- $res[mask, v] = \begin{cases} 0, for\ S \\ inf \end{cases},$
  $mask$ describes whether a vertex is in the path or not.
- Result is in $res[111 \ldots 1, 0]$



Example: Hamiltonian path (TSP)

- Hamiltonian path – graph that visits each vertex exactly once
- Idea: solve it for subgraphs
- $res[mask, v] = \begin{cases} 0, for\ S \\ \quad inf \end{cases}$
- Result in $res[111 \ldots 1, 0]$
- $res[m, v] = \min(res[m, v],$ $res[11\textcolor{red}{0} \ldots 1, i] + w(i, v))$

I'th mask element



Resources

- [1] Introduction to Algorithms, Thomas H. Cormen, chapter 15
- [2] Universal sequential search problems
  http://www.cs.bu.edu/fac/lnd/dvi/ppi9-115.pdf
- [3] Nice article on dynamic programming (rus)
  https://habr.com/ru/post/191498/
- [4] MIT open courseware on DP: course
- [5] More on DP on mccme:
  https://informatics.mccme.ru/course/view.php?id=9

BACKUP