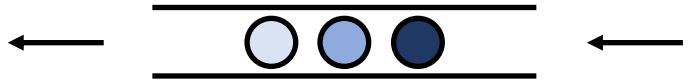
Search Algorithms

Cong Chen

0. Reviewing data structures

Queue (First In First Out, FIFO)

Stack (Last In First Out, LIFO)



Priority Queue (Best Out, Heap)

key	value
314	1.5
926	5.3
…	…

Dictionary (Hash Table)

Configurational Search Strategies

SPRUCE'S
AMERICAN DINING

STARTERS

Honey Glazed Onion Rings
sweet vidalia onions, deep fried in beer batter, glazed with thyme honey 6.50

Smoked Chicken Quesadilla
with caramelized onions, roasted poblano guacamole, jicama salsa & chipotle 8.50

Ancho Chile Shrimp Tacos
with mango salsa, jalapeno-lime crème fraîche, guacamole & shredded cabbage 8.50

Fried Calamari
with housemade marinara and salad greens 8.75

Antipasto Platter
fresh mozzarella, heirloom tomatoes, fresh basil, egg, prosciutto, balsamic vinaigrette 9.25

Sweet Potato Fries
hand cut fries with honey dijon dipping sauce 6.50

SALADS

served with a garlic breadstick

House Salad
with greens, olives, green & red peppers, onions, cucumbers & tomato in orange miso vinaigrette 6.75

Caesar Salad
romaine, fresh parmesan, seasoned croutons & black pepper with house-made garlic anchovy dressing 8.50

Blue Chicken Salad
mesclun greens, apple, grilled chicken, gorgonzola cheese, balsamic vinaigrette 9.25

Red Iceberg Salad
with sweet corn, black berries, goat cheese & fresh basil 9.25

Seared Ahi Tuna Salad
with field greens, cucumber, carrots, tomatoes & crispy onion rings in miso-wasabi dressing 13.00

SANDWICHES

served with house salad, french fries, or potato chips

Pulled Pork
with tangy barbecue sauce on an onion knot 9.50

Turkey Club
roasted turkey breast, bacon, lettuce, avocado & tomato on baguette with aioli 8.00

Reuben
corned beef, melted swiss, sauerkraut & thousand island on marbled rye 7.95

Chicken Salad Croissant
house-made chicken salad with lettuce & tomato on a warm, flaky croissant 7.25

Curried Turkey Wrap
smoked turkey, mozzarella, black olives & curried mayo in a curried spinach wrap 7.75

Shrimp Cilantro Wrap
shrimp, avocado, mixed greens, salsa, cilantro & may on a sun dried tomato tortilla 8.50

BURGERS

served with house salad, french fries, or potato chips

Grass-Fed Beef Burger
with sharp cheddar, heirloom tomatoes & caramelized onions 9.25

Mushroom Swiss Burger
with sautéed mushrooms & melted swiss on a homebaked roll 10.00

Hickory Burger
topped with cheddar, hickory smoked bacon and smoky barbecue sauce 10.00

Black Bean Veggie Burger
house-made veggie burger with tomatoes, avocado & sprouts on a home-baked roll 9.50

ENTREES

served with house salad or cup of soup

Microbrew-Battered Halibut
with sweet potato fries, house made slaw & tartar sauce 13.25

Braised Boneless Short Ribs
with garlic mashed potatoes, sautéed broccolini, gremolata & chipotle aioli 16.00

Grilled Pork Tenderloin
with steamed jasmine rice, red Thai curry glaze & coriander emulsion 15.75

Pistachio Crusted Salmon
with fresh berry salsa, wild rice, potato cake & sautéed snap peas 17.00

Lobster Mac & Cheese
bay shrimp and chunks of Maine lobster baked with gruyere, cheddar, and cotija cheese 15.00

Sauteed Chicken & Shrimp
organic chicken breast and bay shrimp in apricot glace with sage & walnuts 17.95

Homemade Meatloaf
topped with mushrooms and onions, served with steamed fresh vegetables 12.50

Turkey Pot Pie
made with all breast meat, carrots, potatoes, peas, and green beans in a flaky crust 13.00

Pork Chops
broiled in honey dijon sauce, served with herbed mashed potatoes 14.25

SIDES

French Fries *2.50*

Sweet Potato Fries *3.50*

Collard Greens *3.00*

Cheesy Mashed Potatoes *2.00*

Beef Chili *3.75*

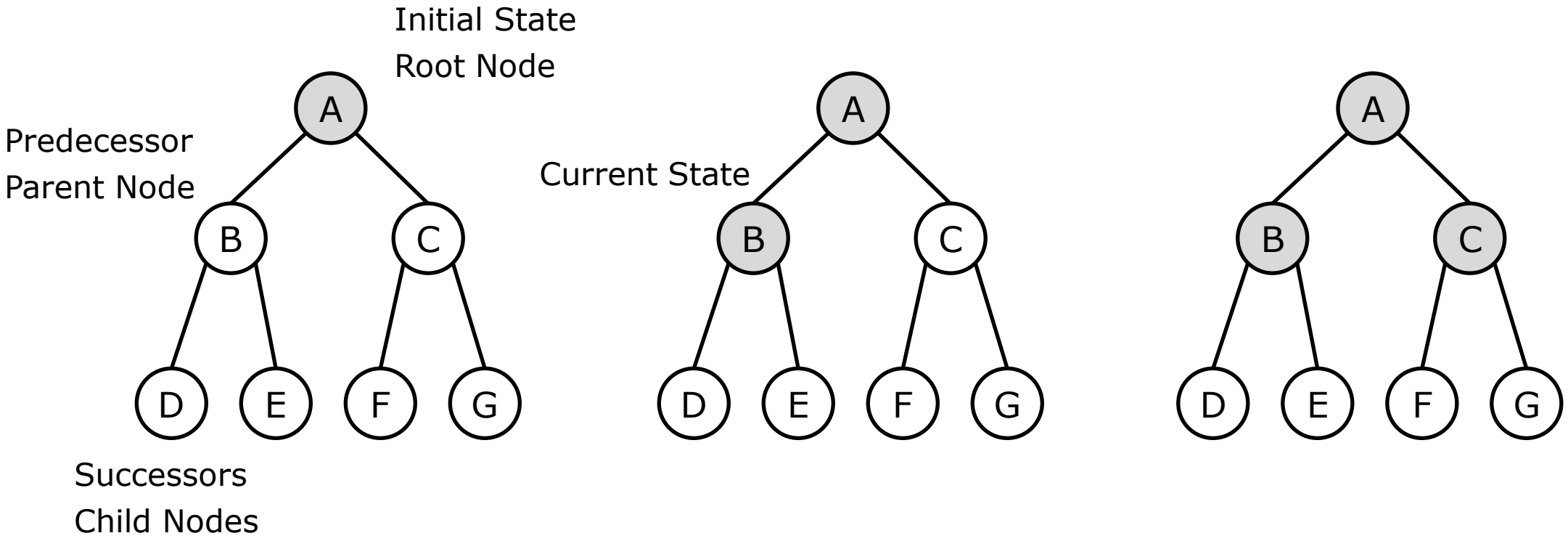
Please alert your server to any special dietary needs.

In a restaurant: Starter? Entrée? Medium-Rare? Sides? Drinks? Dessert?

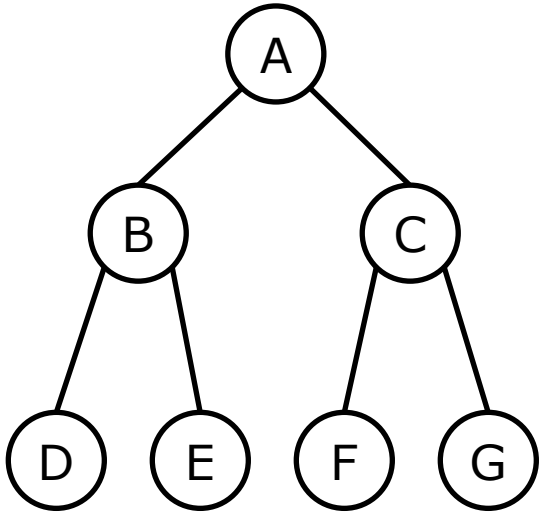
Deciding a search strategy: Depth or Best? One solution or M solutions? Heuristics? Tree? And-or?

1. Traversal

Search algorithms all share this basic structure; they vary primarily according to how they choose which state to expand next—the so-called **search strategy**.



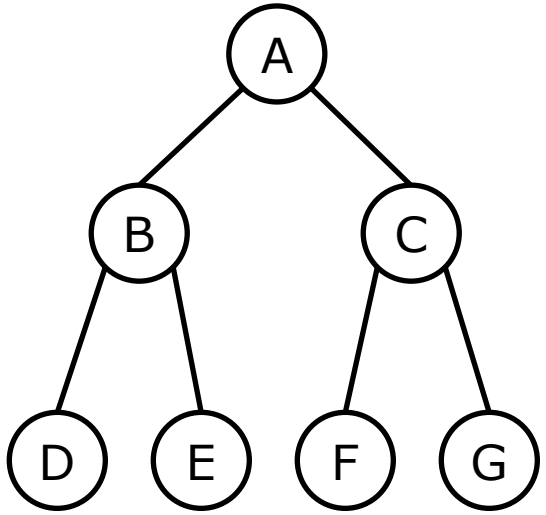
Breadth First Traversal



Step	Queue
0	
1	**A**
2	**B** C
3	**C** D E
4	**D** E F G
5	**E** F G
6	**F** G
7	**G**
8	

```
Q.push(init);

while ( not Q.empty() )
{
     cur = Q.pop();
     Q.push( cur.succs() );
}
```

Depth First Traversal

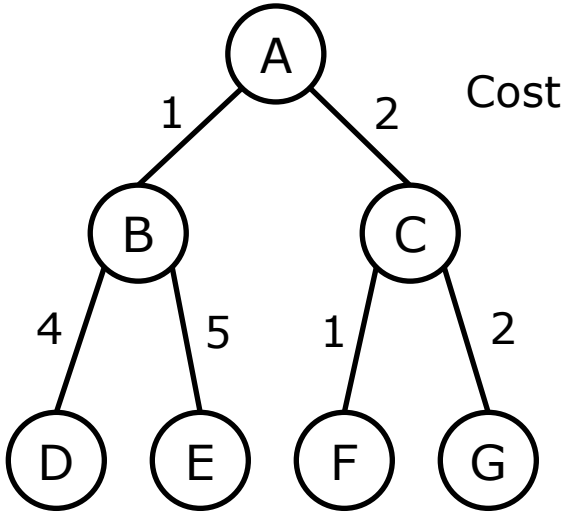


Step	Stack
0	
1	**A**
2	B **C**
3	B F **G**
4	B **F**
5	**B**
6	D **E**
7	**D**
8	

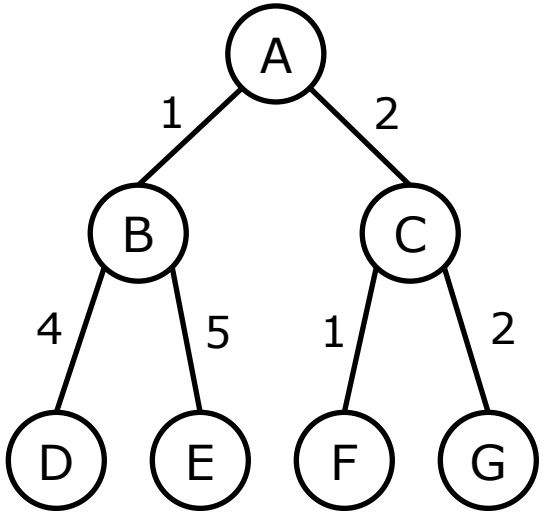
Memory-efficient

```
T.push(init);

while ( not T.empty() )
{
     cur = T.pop();
     T.push( cur.succs() );
}
```

Informed Traversal



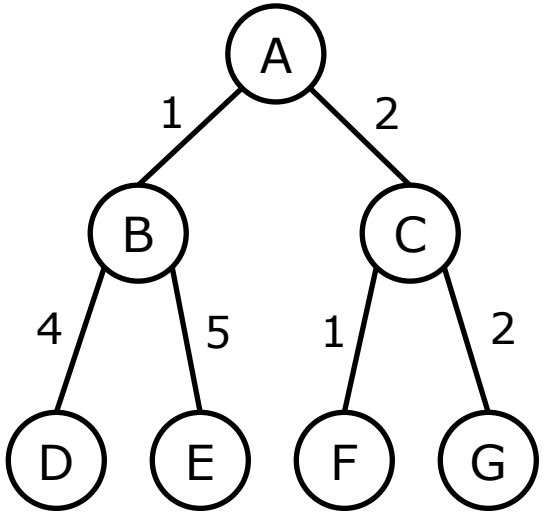
Best First Traversal



Step	Priority Q
0	
1	**A⁰**
2	**B¹** C²
3	**C²** D⁵ E⁶
4	**F³** G⁴ D⁵ E⁶
5	**G⁴** D⁵ E⁶
6	**D⁵** E⁶
7	**E⁶**
8	

```
PQ.push(init);

while ( not PQ.empty() )
{
     cur = PQ.pop();
     PQ.push( cur.succs() );
}
```

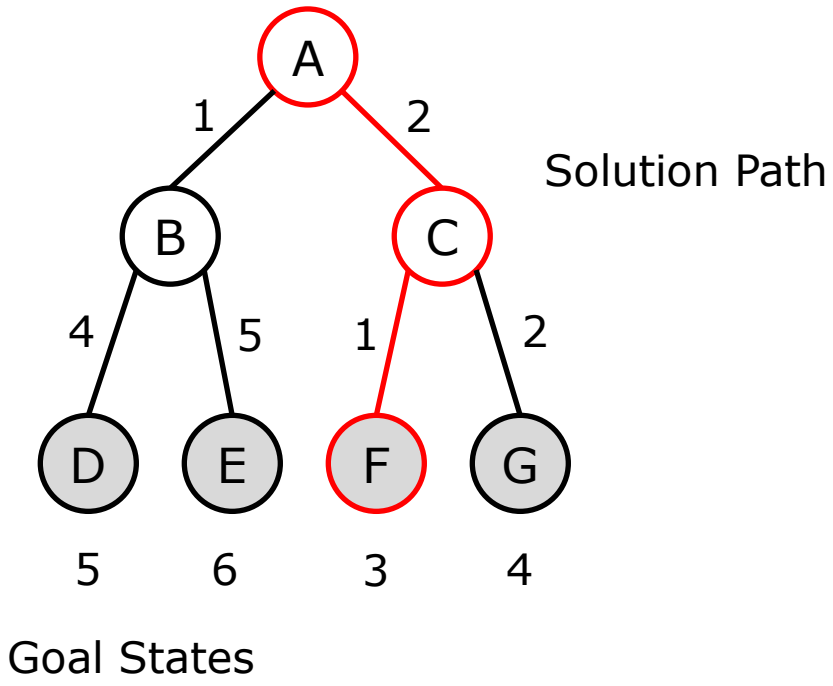
Depth First Traversal +



Step	Stack
0	
1	**A^0**
2	C^2 **B^1**
3	C^2 E^6 **D^5**
4	C^2 **E^6**
5	**C^2**
6	G^4 **F^3**
7	**G^4**
8	

```
T.push(init);

while ( not T.empty() )
{
     cur = T.pop();
     T.push(cur.succs().sort(R));
}
```

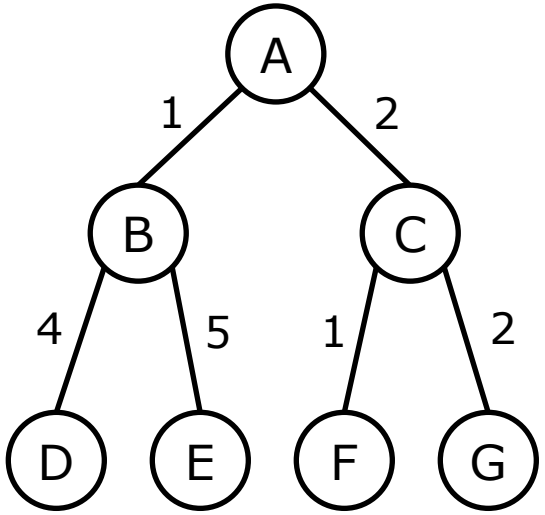
2. Search Tasks: 1-Best Solution VS M-Best Solutions



1-Best Solution: Find the best one from the goal state set

M-Best Solutions: Find top M best from the goal state set

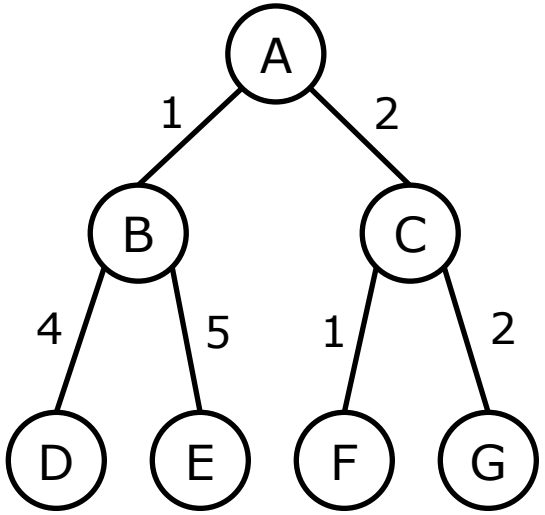
Best First Search – 1-Best



Step	Priority Q
0	
1	**A⁰**
2	**B¹** C²
3	**C²** D⁵ E⁶
4	**F³** G⁴ D⁵ E⁶
~~5~~	~~**G⁴** D⁵ E⁶~~
~~6~~	~~**D⁵** E⁶~~
~~7~~	~~**E⁶**~~
~~8~~	

```
PQ.push(init);

while ( not PQ.empty() )
{
    cur = PQ.pop();

    if ( cur.goal() )
    {
        output(cur);
        break;
    }

    PQ.push( cur.succs() );
}
```

Depth First Traversal – 1-Best



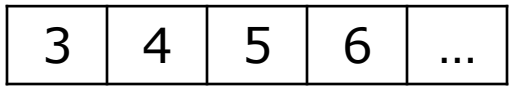
Step	Stack
0	
1	**A^0**
2	C^2 **B^1**
3	C^2 E^6 **D^5**
4	C^2 **E^6**
5	**C^2**
6	G^4 **F^3**
7	**G^4**
8	

```
T.push(init);
best = ∞;

while ( not T.empty() )
{
    cur = T.pop();

    if ( cur.goal() )
    {
        if ( cur < best ) best = cur;
    }

    succs = cur.succs();
    succs.sort(Reverse);
    T.push(succs);
}
output(best)
```

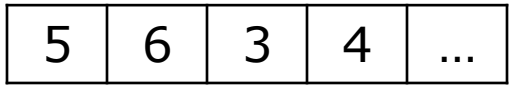
on_goal()

```
PQ.push(init);

while ( not PQ.empty() )
{
    cur = PQ.pop();

    if ( cur.goal() )
        signal = on_goal(cur);
    …

    PQ.push( cur.succs() );
}
```

Goal State Stream: (Best First)

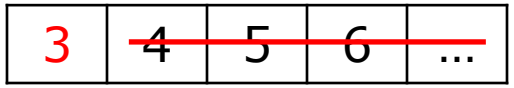


Goal State Stream: (Depth First)



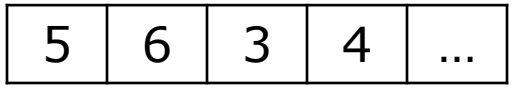
on_goal() – 1-Best

Goal State Stream:
(Best First)

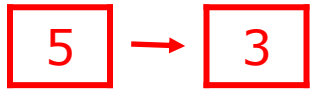


Output the first solution and stop

Goal State Stream:
(Depth First)

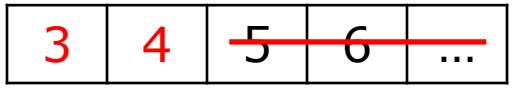


Updating a current best solution



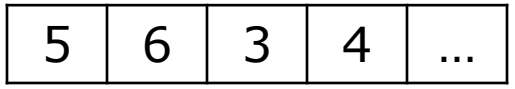
on_goal() – M-Best

Goal State Stream:
(Best First)

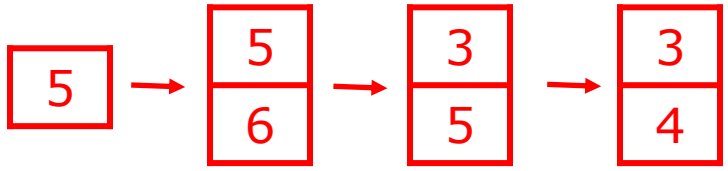


Output the first M=2 solutions and stop

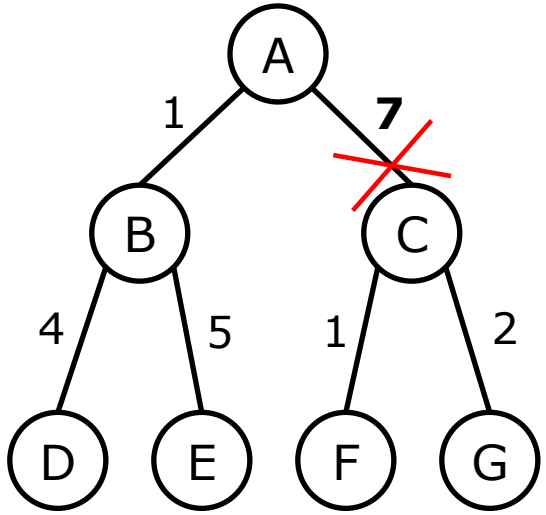
Goal State Stream:
(Depth First)

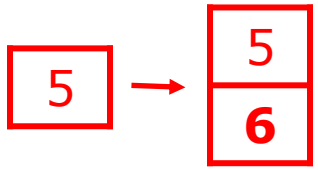


Updating a current best solution list
(max size = 2)



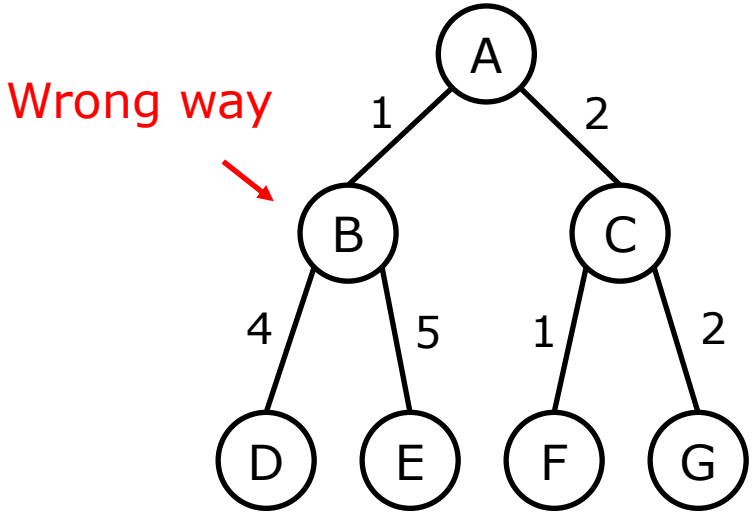
Depth First Pruning





Start pruning when we have already got M=2 solutions

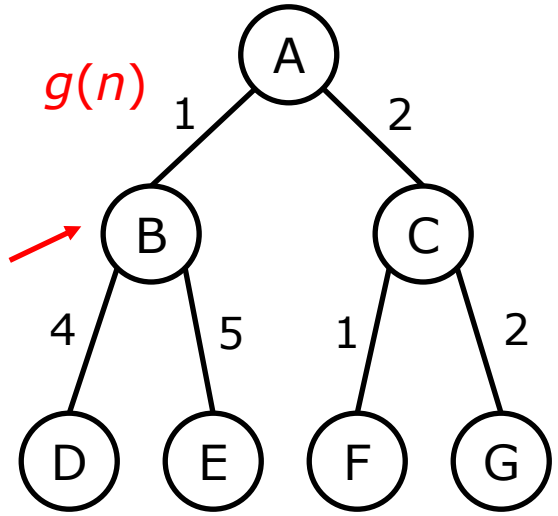
3. Heuristic Functions



Step	Priority Q
0	
1	**A⁰**
2	**B¹** C²
3	**C²** D⁵ E⁶
4	**F³** G⁴ D⁵ E⁶
5	**G⁴** D⁵ E⁶
6	**D⁵** E⁶
7	**E⁶**
8	

Heuristic Functions (Addition)

Estimate at least 4 in the future: $h(n)$

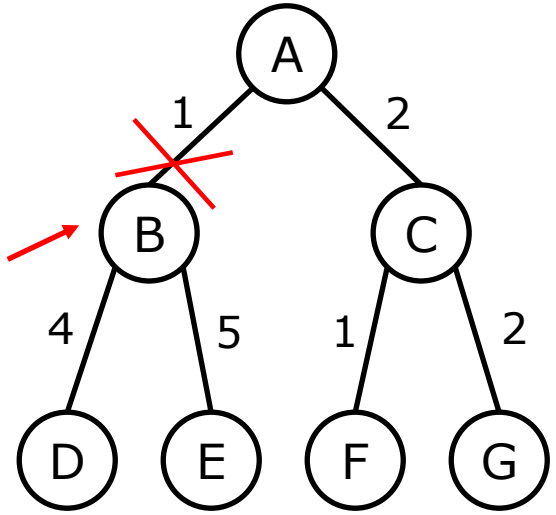


Step	Priority Q
0	
1	**A^{0}**
2	**C^{2+0}** B^{1+4}
3	…

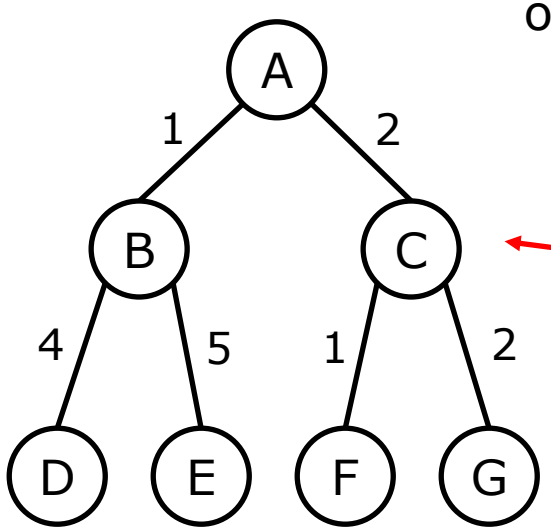
$f(n) = g(n) + h(n)$

Heuristics for pruning

Estimate at least 4 in the future: $h(n)$



Admissible Heuristics

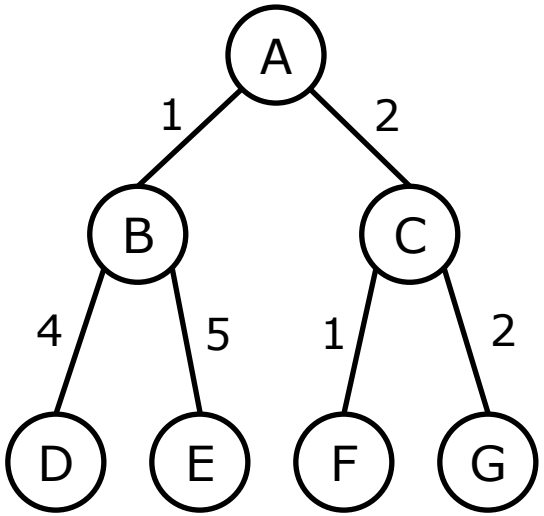


An **admissible** heuristic never overestimates the cost to reach the goal

Can we say "At least 4"?

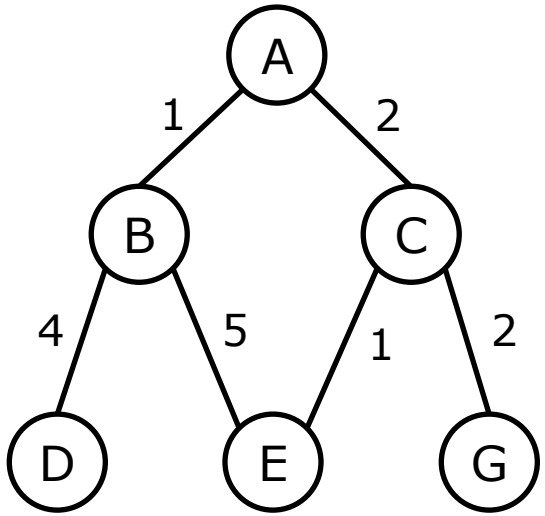
4. Graph Search Space

> *"Those that fail to learn from history are doomed to repeat it."*
>
> –– Winston Churchill



Tree

No loops

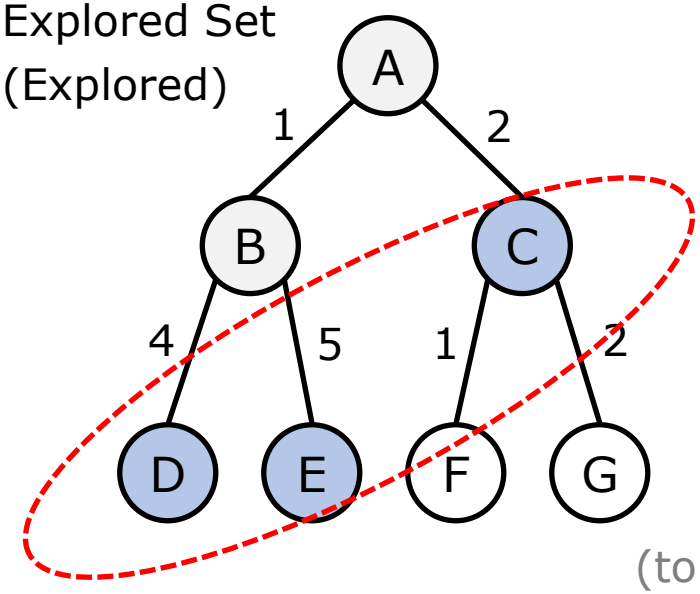


Graph

have loops

Open List and Closed List

Closed List
Explored Set
(Explored)

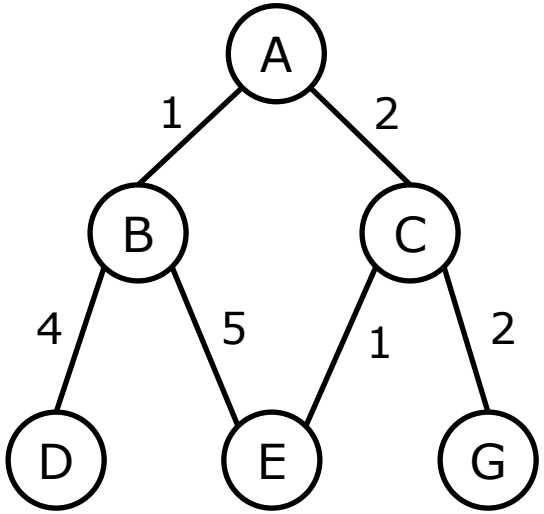


Open List
Frontier
(Exploring)

(to Explore)

Step	Priority Q
0	
1	**A⁰**
2	**B¹** C²
3	**C²** D⁵ E⁶
4	**F³** G⁴ D⁵ E⁶
5	**G⁴** D⁵ E⁶
6	**D⁵** E⁶
7	**E⁶**
8	

Open List and Closed List

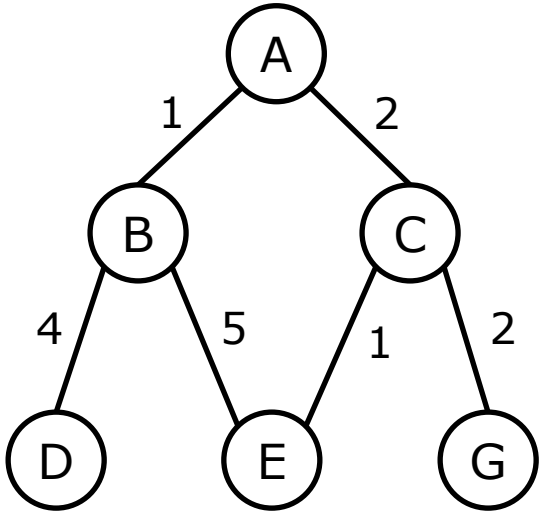


Step	Open
0	
1	**A^0**
2	**B^1**C^2
3	D^5**E^6**C^2
4	D^5**C^2**
5	$D^5E^3G^4$

Closed					
A	B	C	D	E	G
0					
0	1				
0	1			6	
0	1	2			
...					

Put back to Open

Open List and Closed List



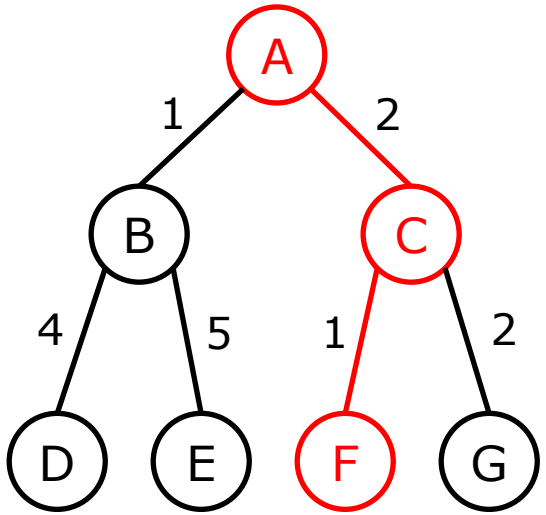
Step	Open
0	
1	**A^0**
2	**B^1**C^2
3	D^5**E^6**C^2
4	D^5**C^2**
5	$D^5E^3G^4$

Open + Closed					
A	B	C	D	E	G
∞	∞	∞	∞	∞	∞
0	∞	∞	∞	∞	∞
0	1	2	∞	∞	∞
0	1	2	5	6	∞
0	1	2	5	6	∞
0	1	2	5	3	∞
…					

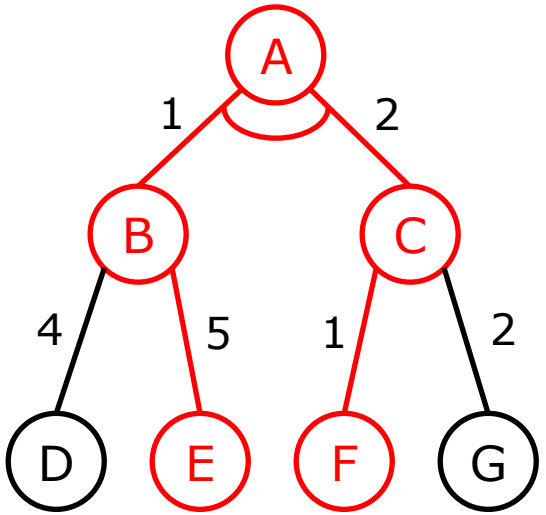
Search Strategies

- Dijkstra = Best First Graph Search
- A* = Dijkstra + Heuristics (addition)
- Depth First Branch and Bound (DFBnB) = Depth First + Heuristics + Pruning

5. And-Or Search Space

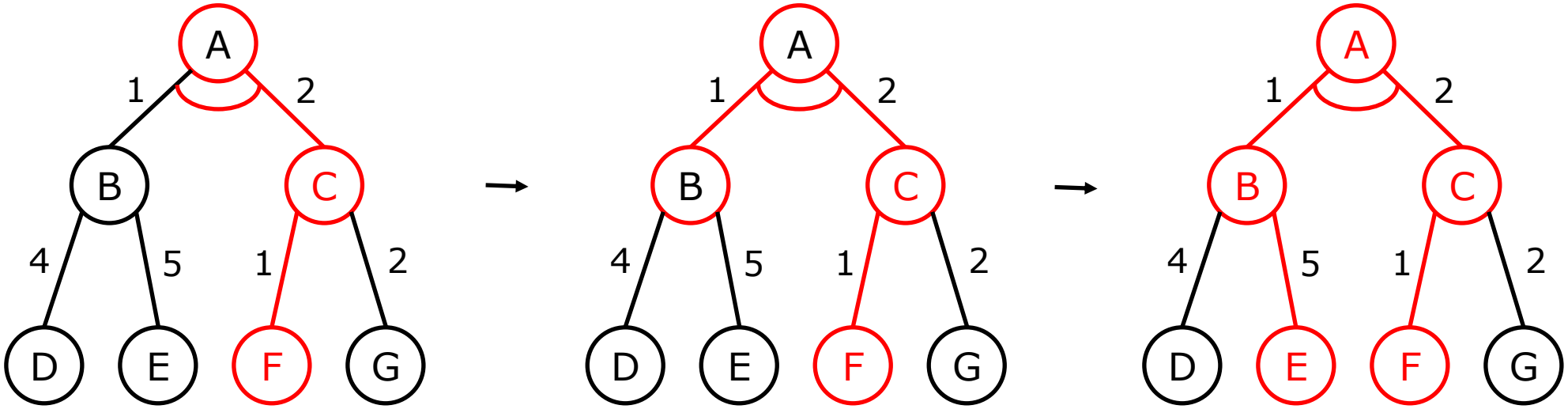


Or Search



And-Or Search

Labeling Procedure



Configurational Search Strategies

Deciding a search strategy:

Depth or Best?

One solution or M solutions?

Heuristics? Pruning?

Tree or Graph?

And-or?