Best First Search

A* Algorithm

Notion of Heuristics

- **Heuristics use domain specific knowledge to estimate the quality or potential of partial solutions**
- Examples
  - **Manhattan distance heuristic for 8 puzzle**

Calculating Cost

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

g(n) – Actual cost of traversing from initial state to state n

h(n) – Estimated cost of reaching to the goal from state n

Informed State Space

- **Given: [S, s, O, G, h] where**
  - **S is the (implicitly specified) set of states**
  - **s is the start state**
  - **O is the set of state transition operators each having some cost**
  - **G is the set of goal states**
  - **h( ) is a heuristic function estimating the distance to a goal**
- **To find:**
  - Min. cost of sequence of transactions to the goal state

A* Algorithm

1. **Initialize:** Set OPEN = {s}, CLOSE = { }, Set $f(s) = h(s)$, $g(s)=0$
1. **Fail:**
   - If OPEN ={ }, Terminate with Failure
2. **Select:** Select the minimum cost state, n, form OPEN and Save n in CLOSE
3. **Terminate:**
   - If $n \in G$, terminate with SUCCESS

A* Algorithm

5. **Expand:**

   - Generate the successors of n using O. For each successor, m, insert m in OPEN only if $m \notin [OPEN \cup CLOSE]$

     set $g(m) = g(n) + C(n,m)$

     set $f(m) = g(m) + h(m)$

     insert m in OPEN

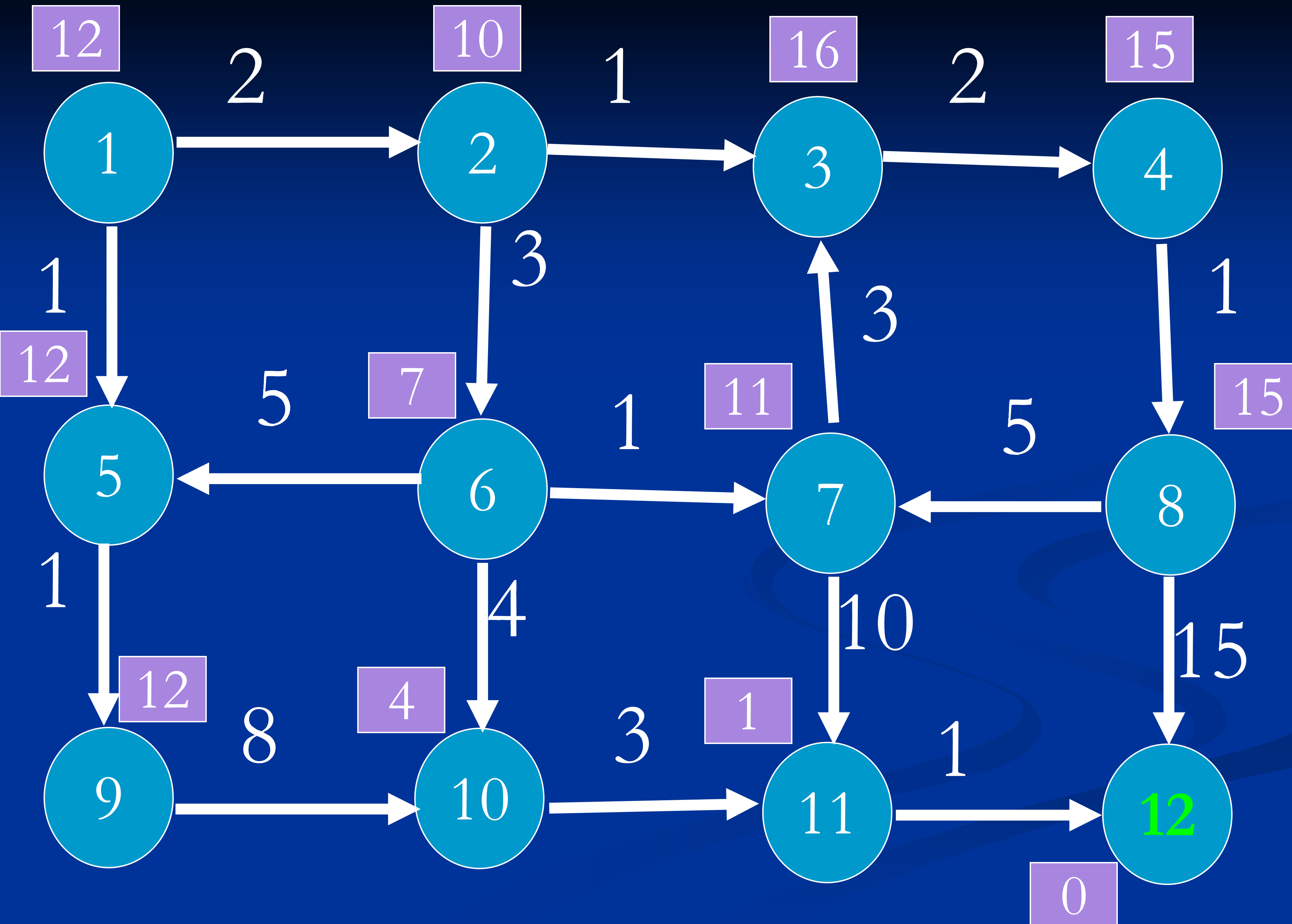
     if $m \in [OPEN \cup CLOSE]$

     Set $g(m) = \min\{ g(m), g(n)+C(m,n)\}$

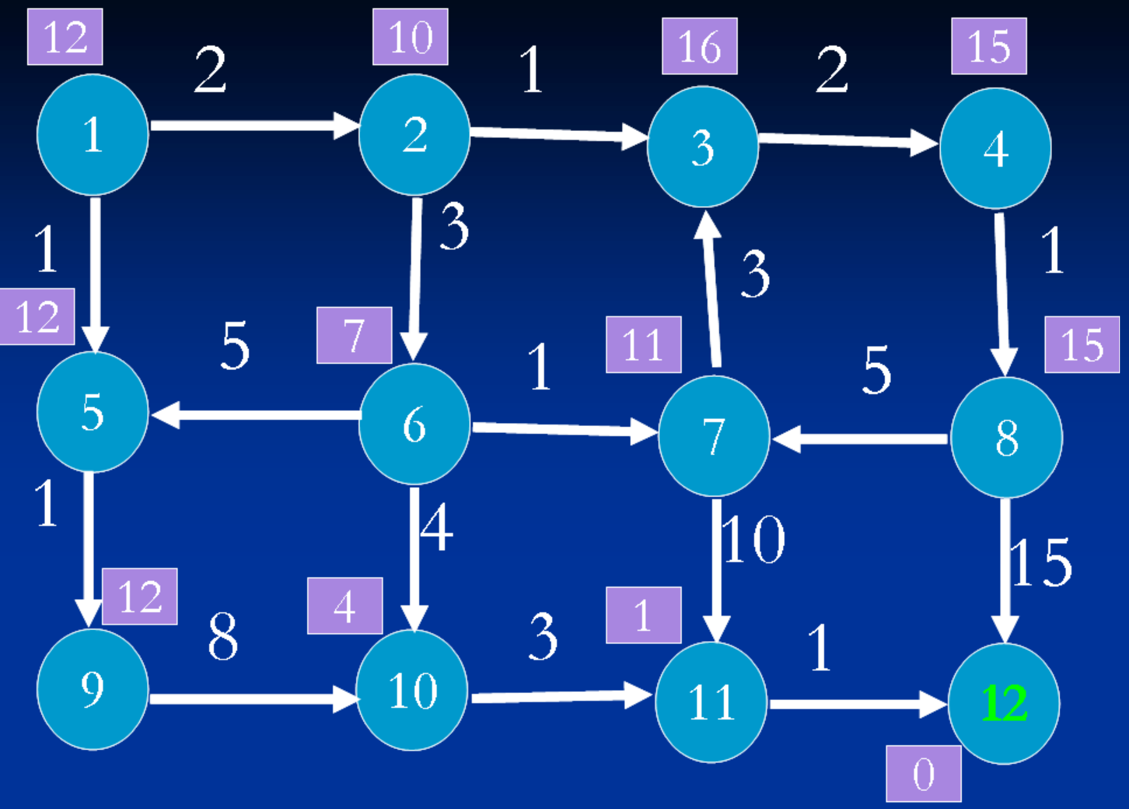
     set $f(m) = g(m) + h(m)$

     If f(m) has decreased and $m \in CLOSE$ move it to OPEN

6. **Loop:** Goto step 2



12
10
16
15
2
1
2
1
2
3
4
3
1
1
3
12
7
11
15
5
1
5
5
6
7
8
1
4
10
15
12
4
1
8
3
1
9
10
11
12
0

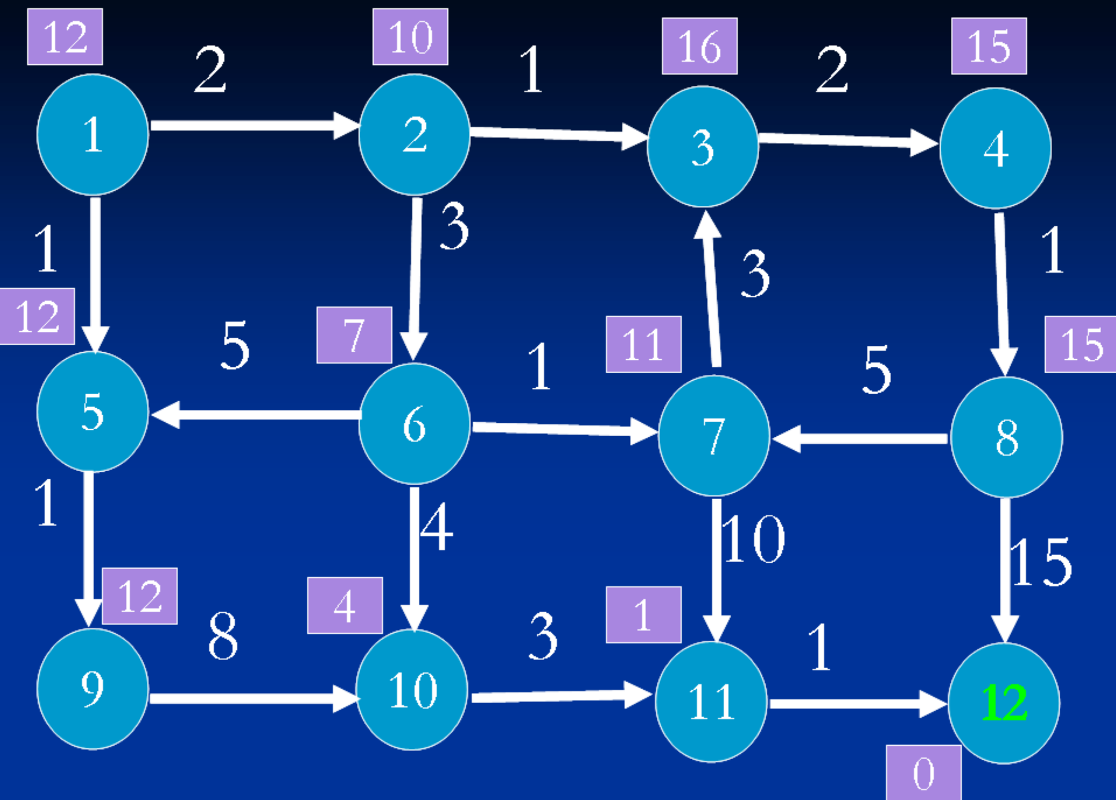


OPEN

1(12)

CLOSE

Node	g()
1	0

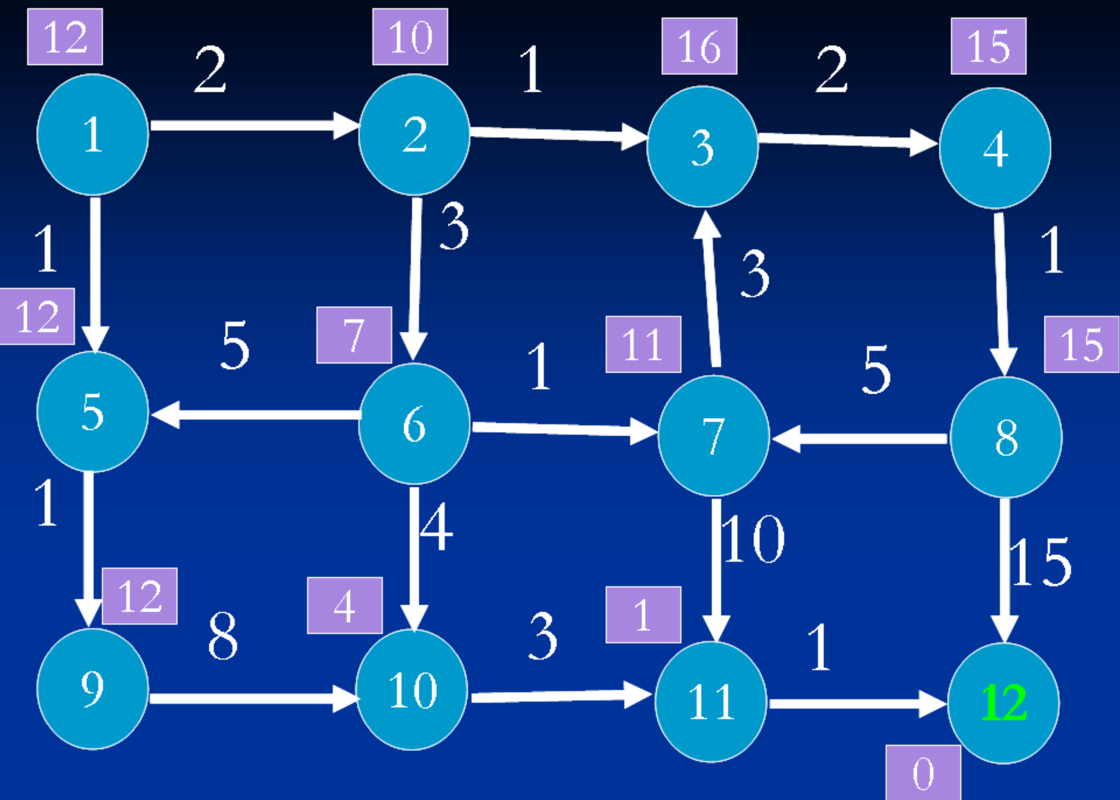


OPEN

Node	g()
1	0

CLOSE

1(12)



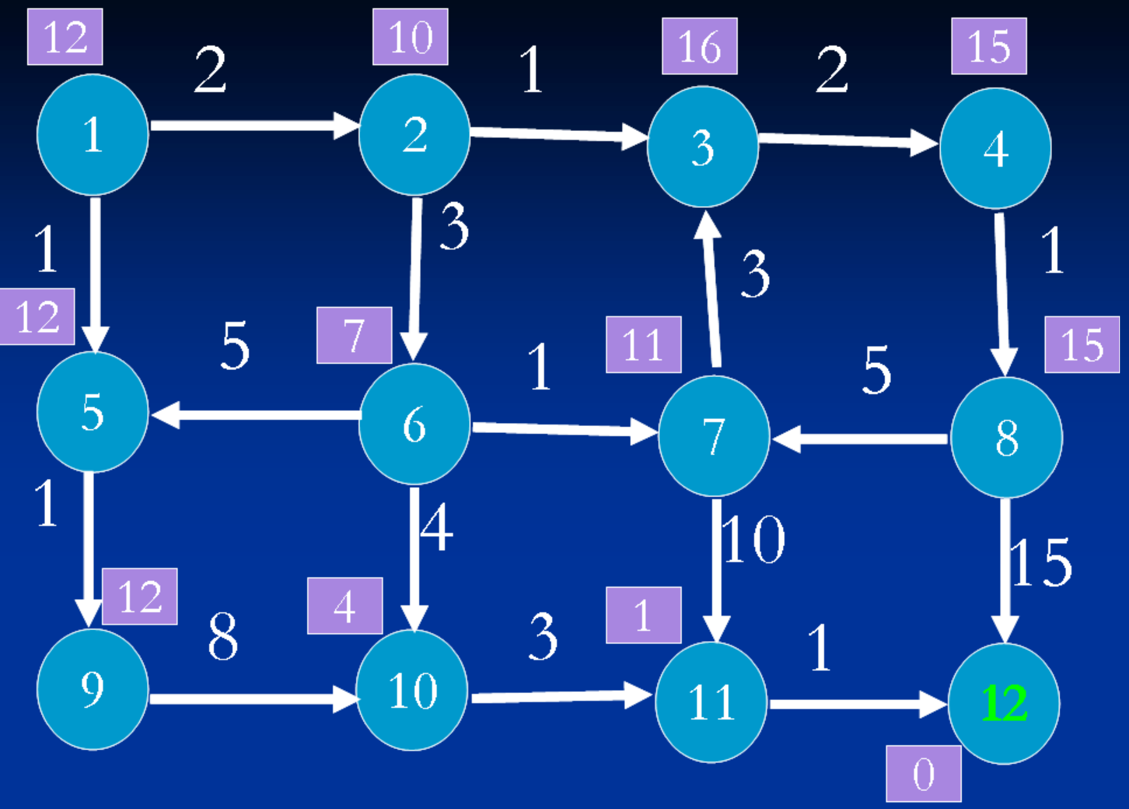
OPEN

2(12) 5(13)

CLOSE

1(12)

Node	g()
1	0
2	2
5	1



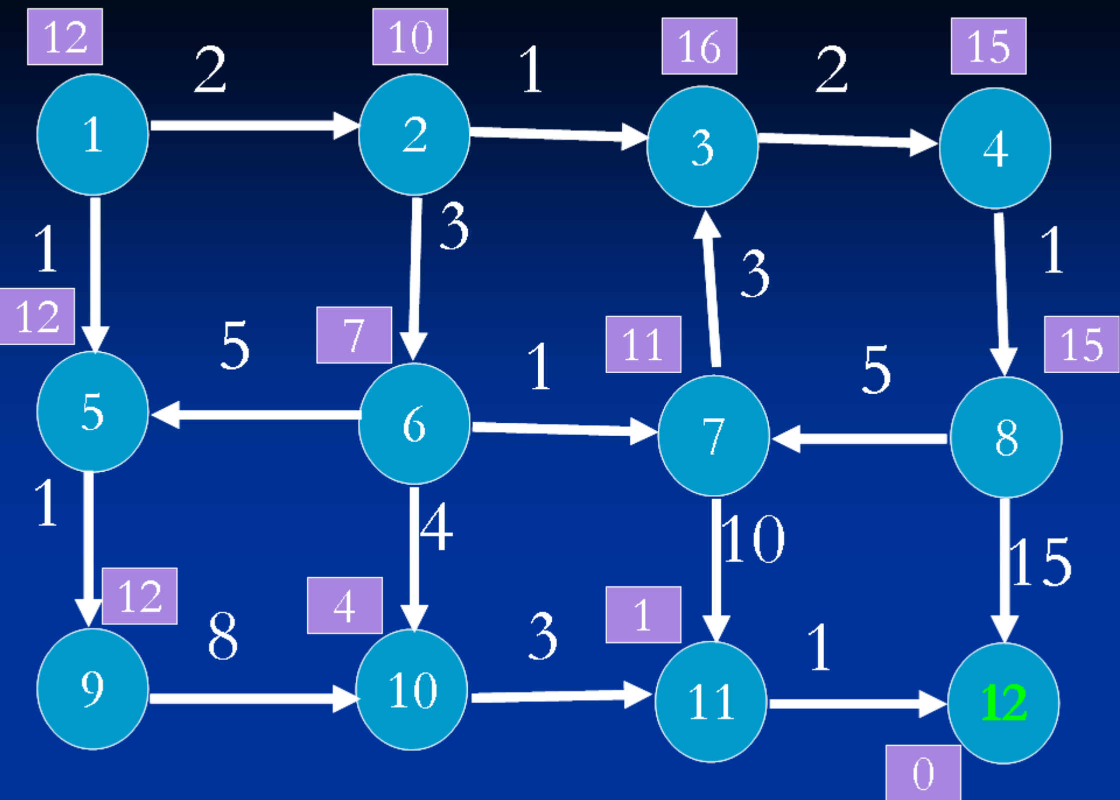
OPEN

5(13)

CLOSE

1(12) 2(12)

Node	g(n)
1	0
2	2
5	1



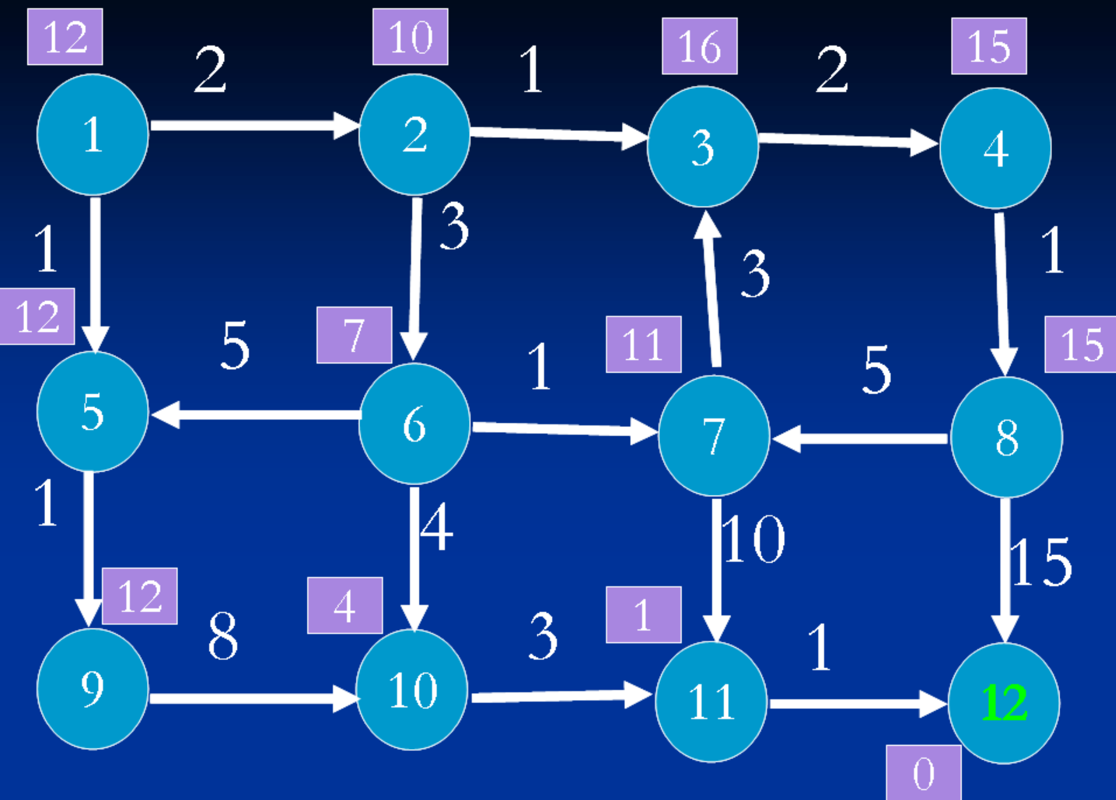
OPEN

5(13) 3(19) 6(12)

CLOSE

1(12) 2(12)

Node	g()
1	0
2	2
5	1
3	3
6	5



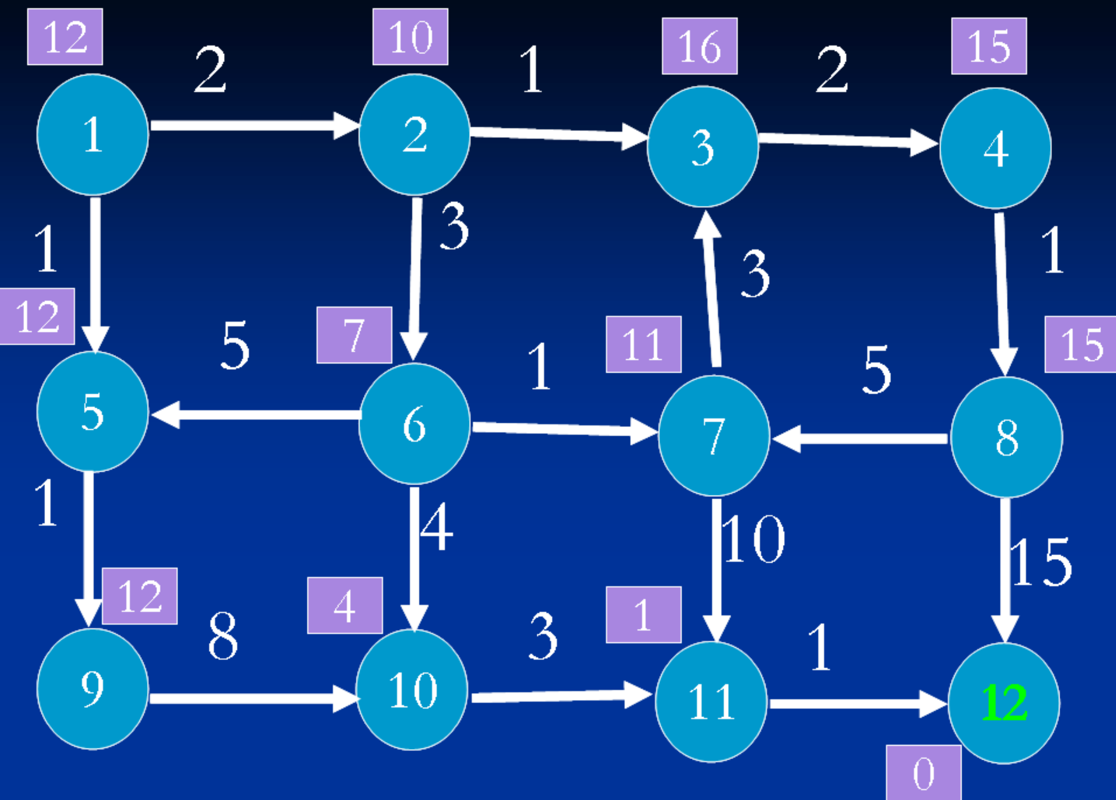
OPEN

5(13) 3(19)

CLOSE

1(12) 2(12) 6(12)

Node	g()
1	0
2	2
5	1
3	3
6	5



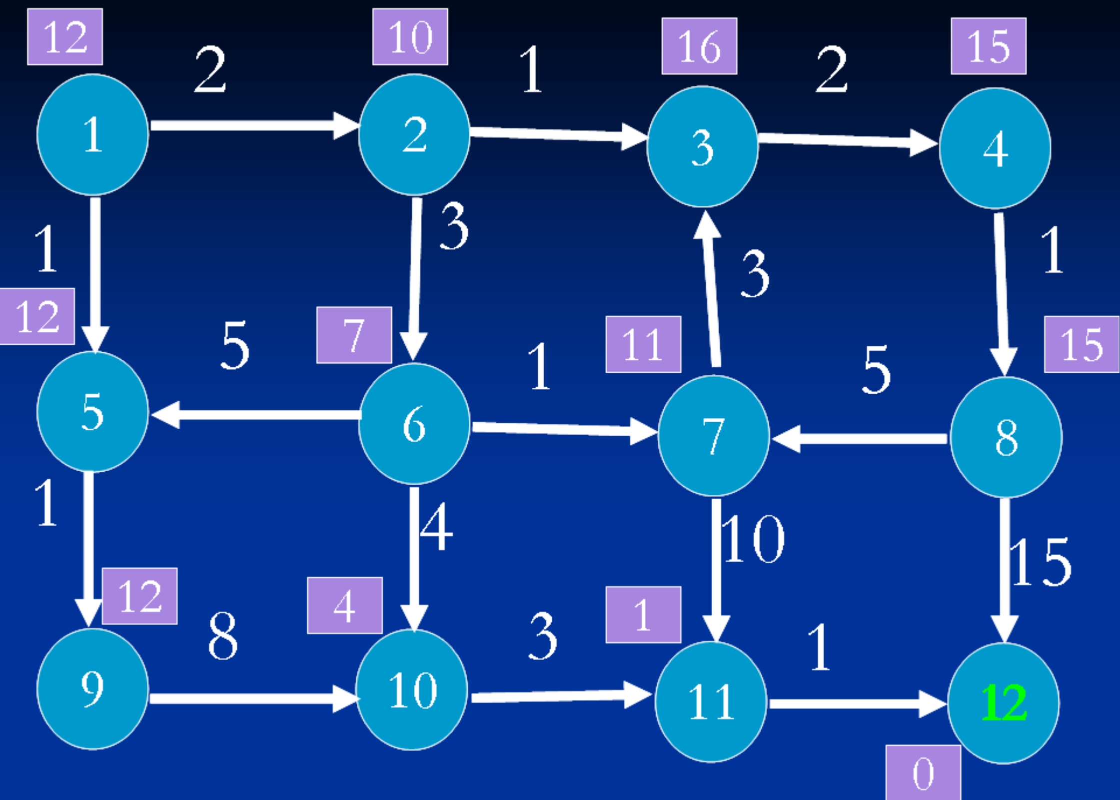
OPEN

10(13) 3(19) 5 (13)
7(17)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9

CLOSE

1(12) 2(12) 6(12)



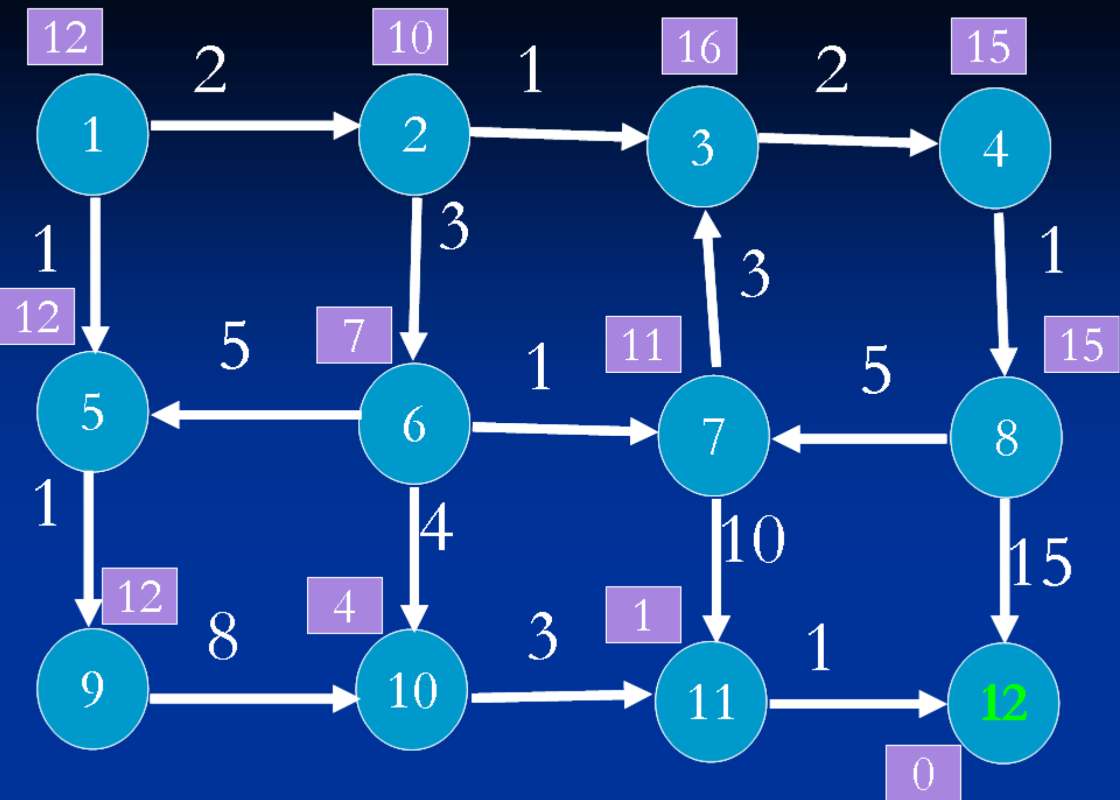
OPEN

10(13) 3(19) 7(17)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9

CLOSE

1(12) 2(12) 6(12) 5(13)



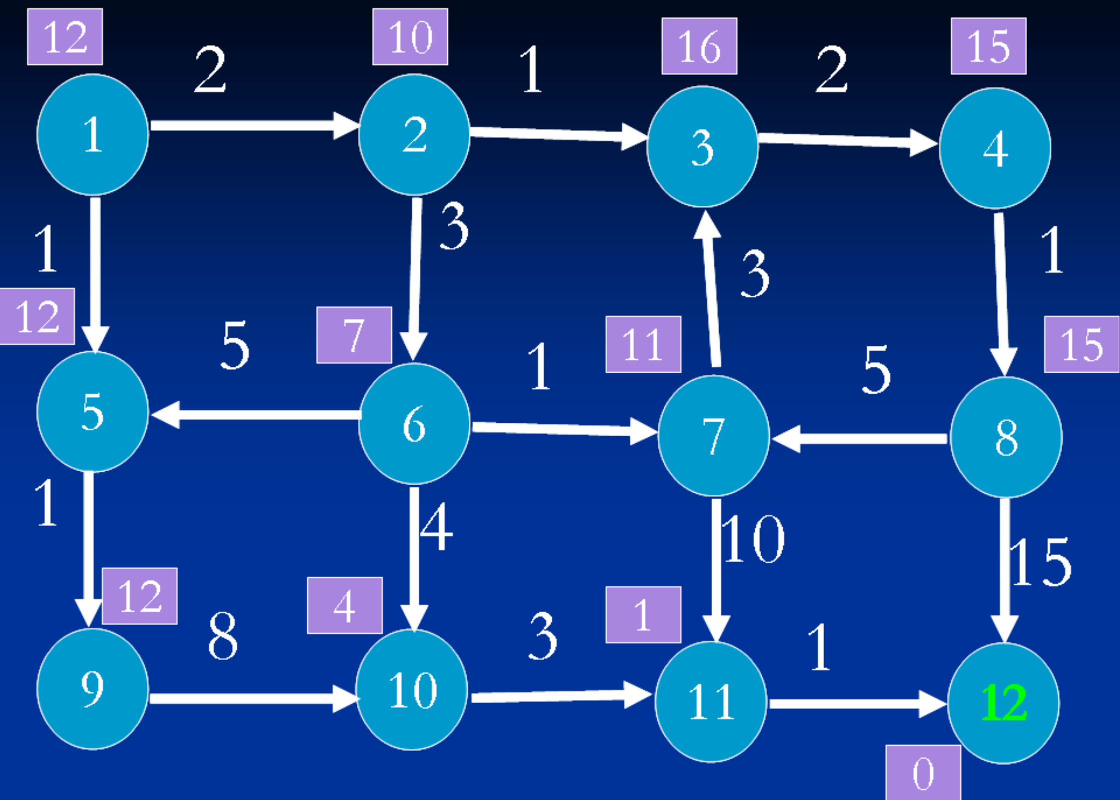
OPEN

3(19) 10(13) 7(17)
9(14)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2

CLOSE

1(12) 2(12) 6(12) 5 (13)



OPEN

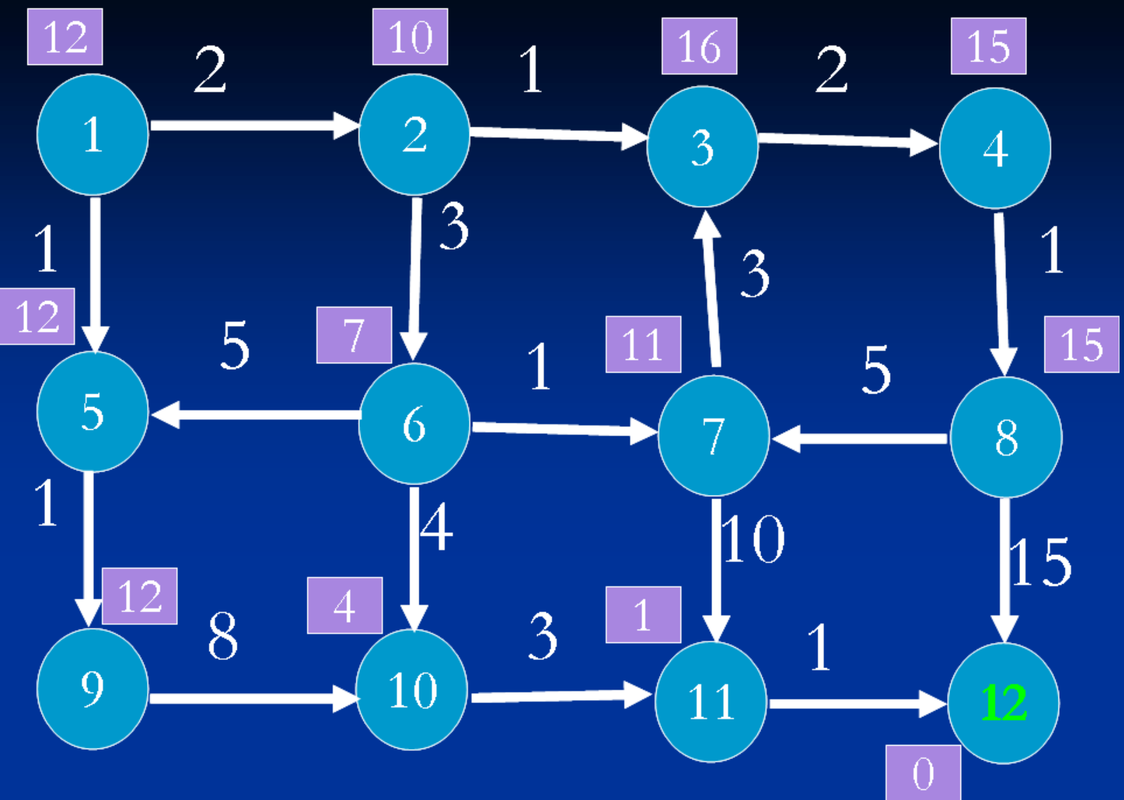
3(19) 7(17) 9(14)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2

CLOSE

1(12) 2(12) 6(12) 5 (13)

10(13)



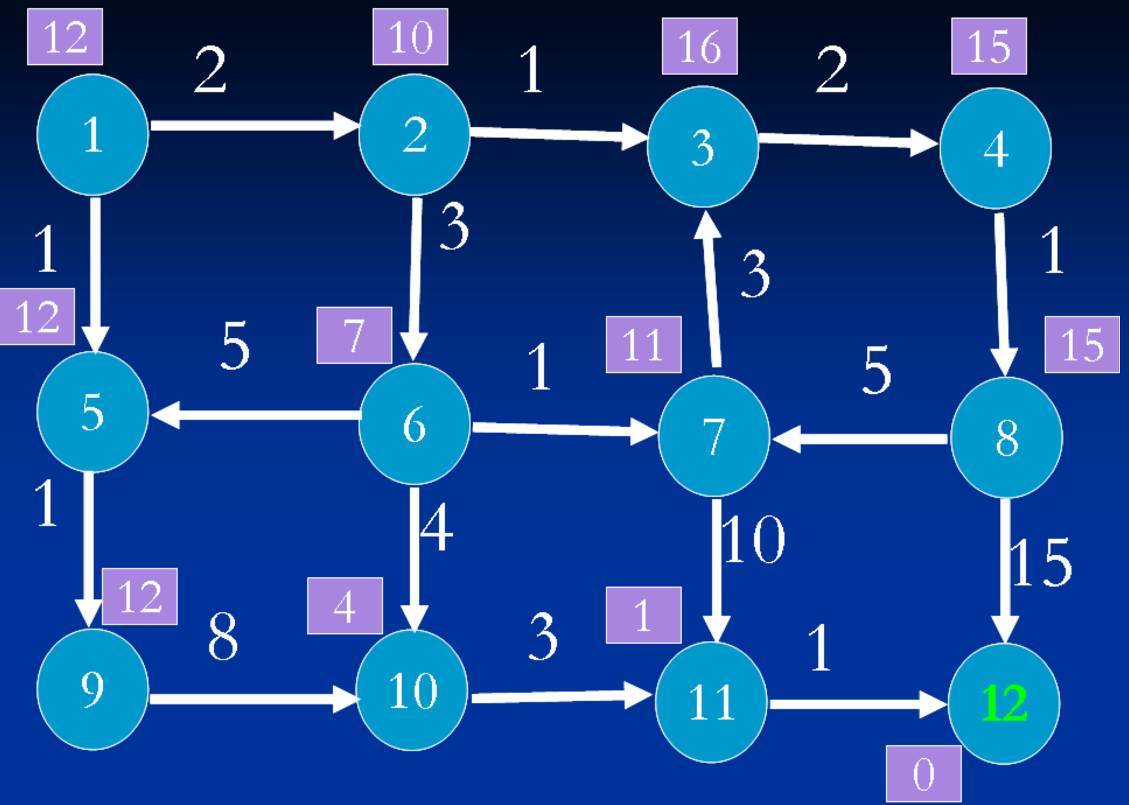
OPEN

3(19) 7(17) 9(14) 11(13)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2
11	12

CLOSE

1(12) 2(12) 6(12) 5 (13) 10(13)



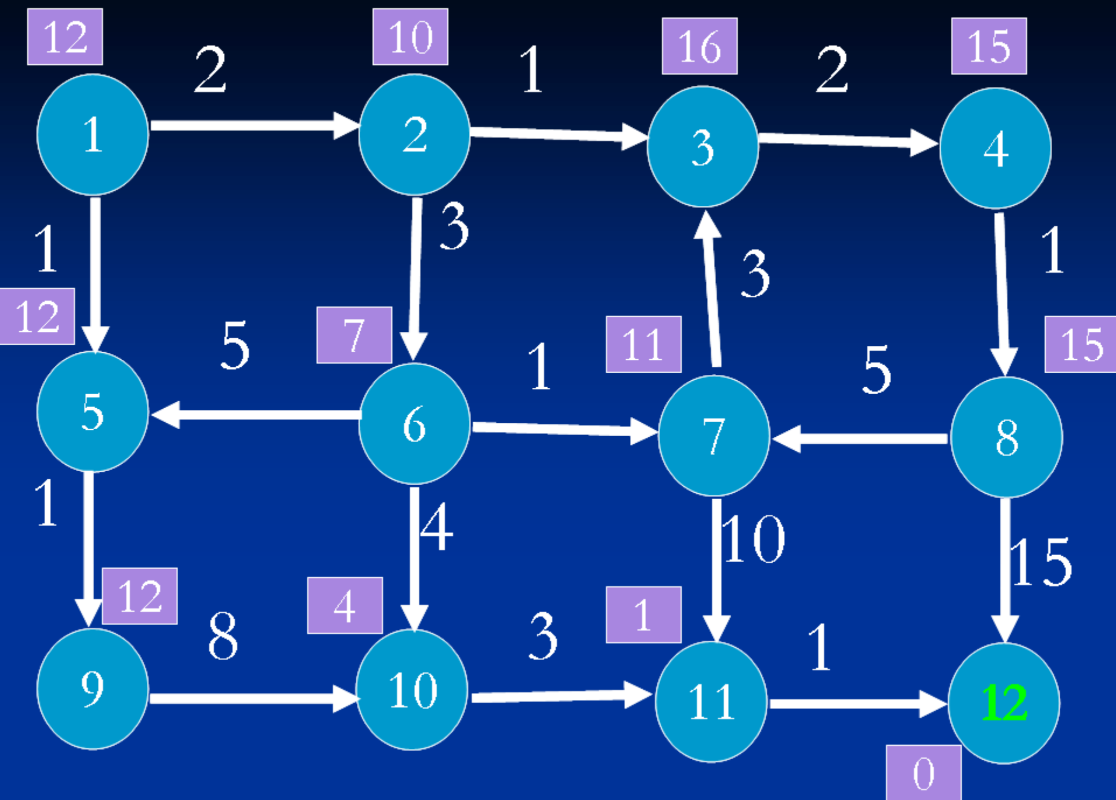
OPEN

3(19) 7(17) 9(14)

CLOSE

1(12) 2(12) 6(12) 5 (13)
10(13) 11(13)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2
11	12



OPEN

3(19) 7(17) 9 (14)

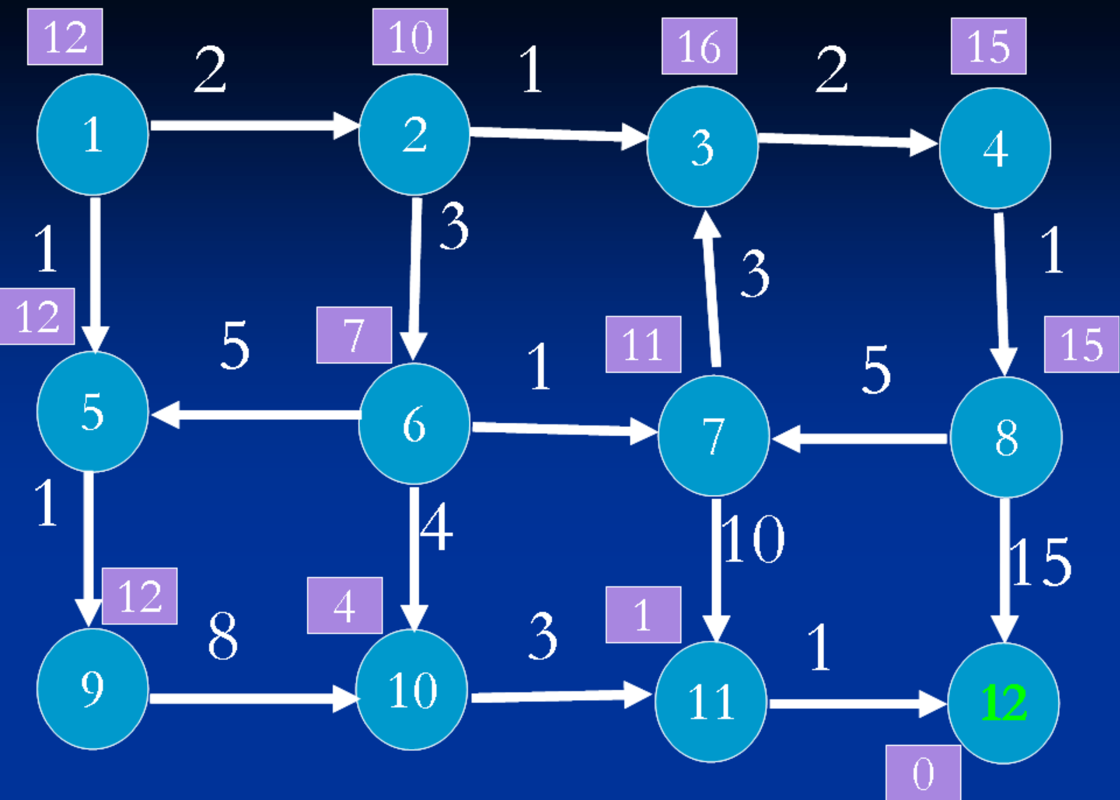
12 (13)

CLOSE

1(12) 2(12) 6(12) 5 (13)

10(13) 11(13)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2
11	12
12	13



OPEN

3(19) 7(17) 9 (14)

Node	g()
1	0
2	2
5	1
3	3
6	5
7	6
10	9
9	2
11	12
12	13

CLOSE

1(12) 2(12) 6(12) 5 (13)
10(13) 11(13) 12 (13)

Comparing with OR Graph Search

- Instead of 11 nodes (OR) we expanded only 7 nodes in A*
- Inference: Nodes which looked promising initially were found to be not so good later on and were ignored/left off

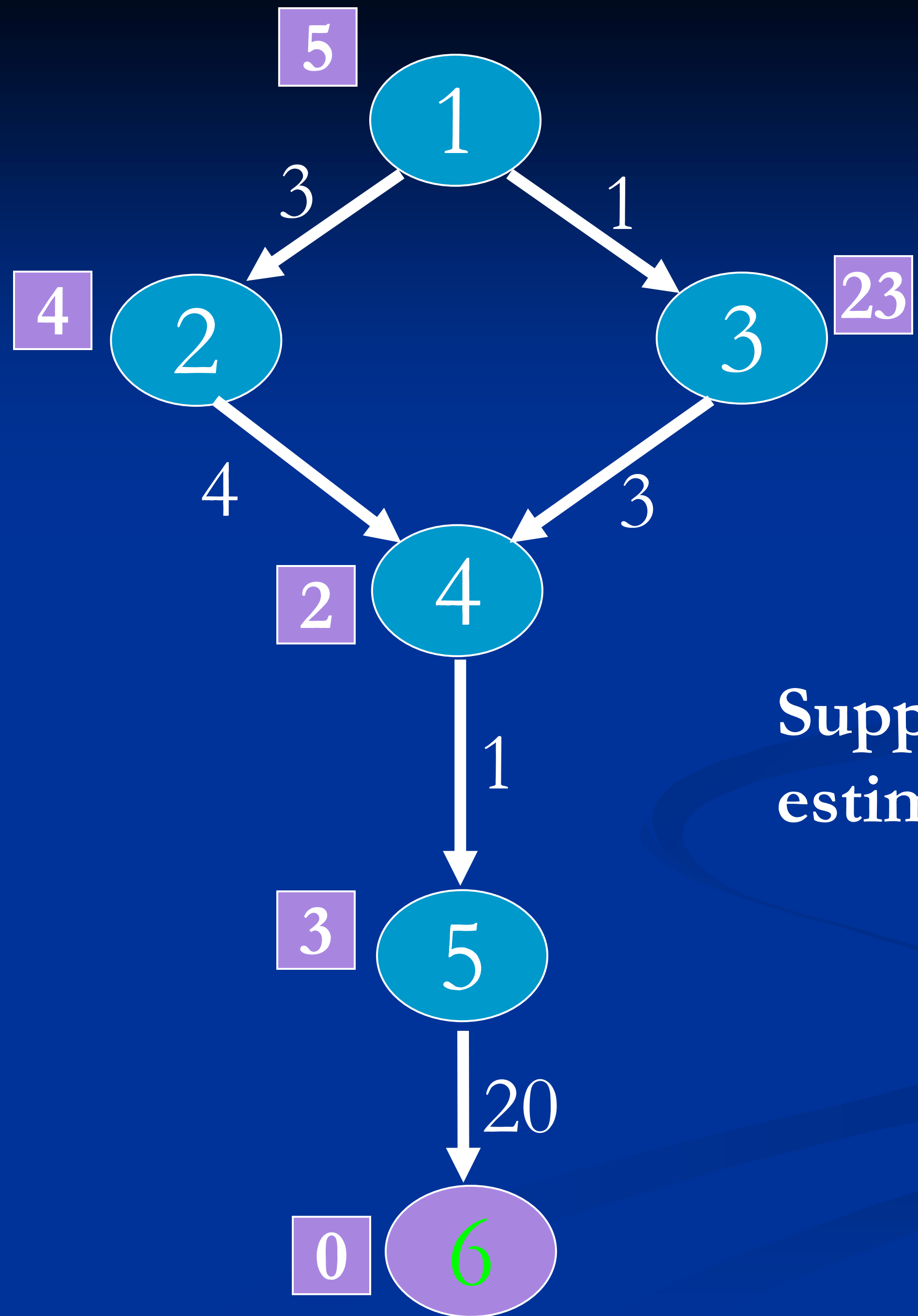
Claims

- If **f(n) < C*** then n must be expanded

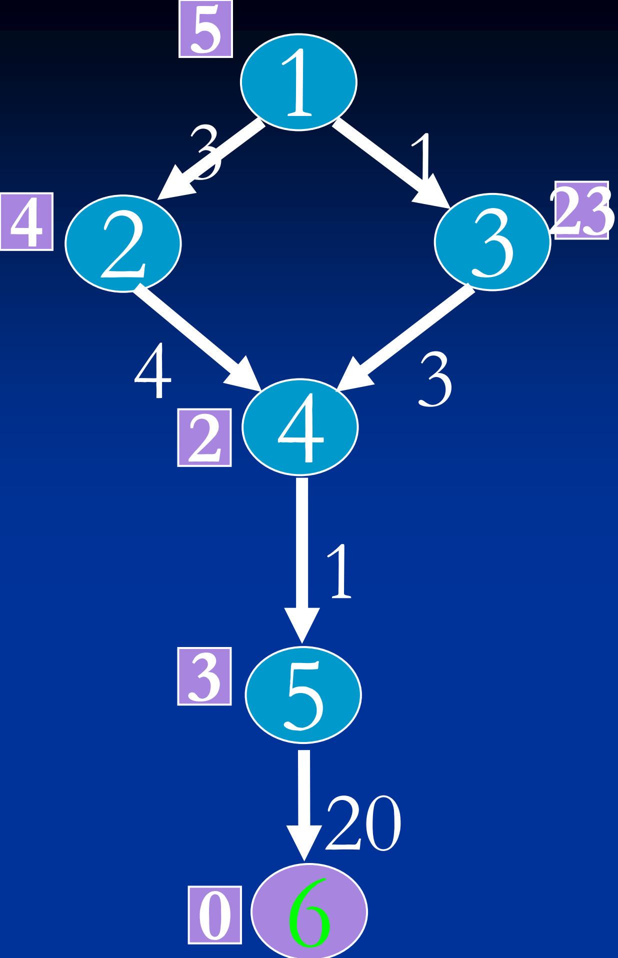
Assumption

1. The heuristic function under estimates

   h(n) <= f*(n) (Cost of reaching goal from n)
2. All costs are +ve
3. If you have nodes with same costs **(f() value)** then select the one which has minimum g() value
4. At times we have to expand sub optimal paths before expanding optimal paths **(Non – monotonicity of Heuristic Function)**



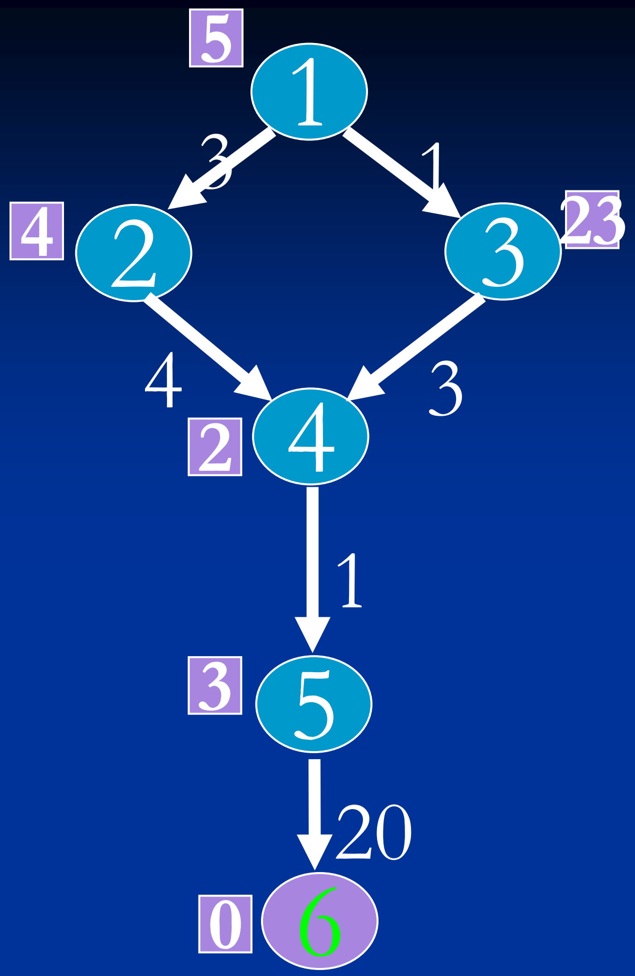
Suppose h() under estimates



OPEN

1(5)

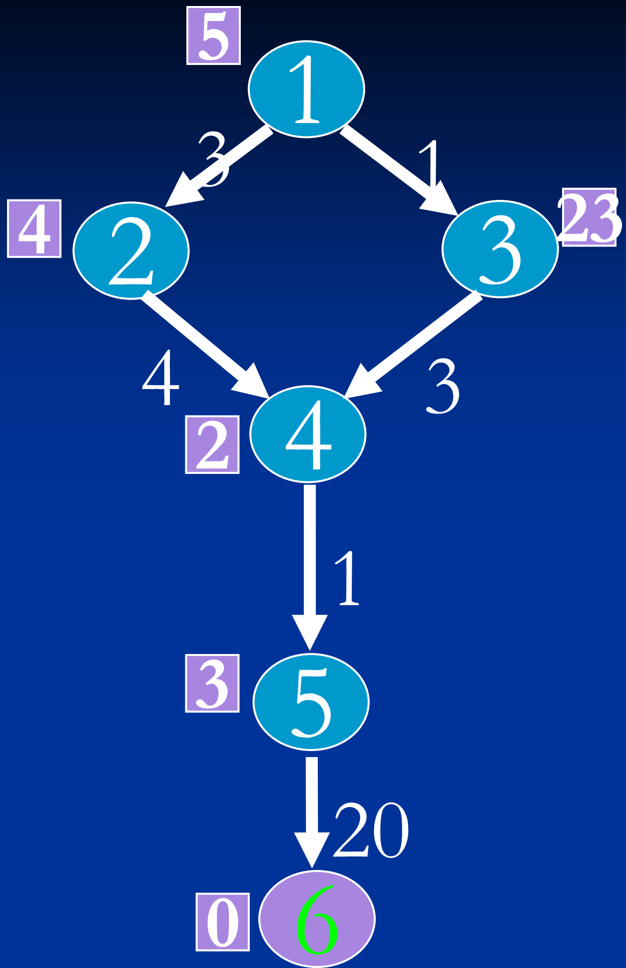
CLOSE



OPEN

CLOSE

1(5)

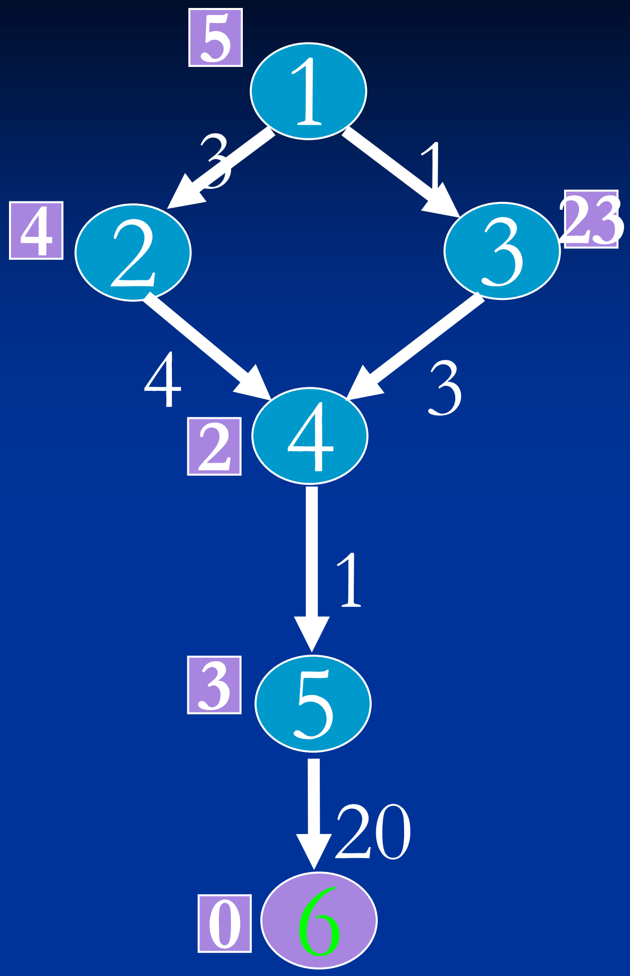


OPEN

2(7) 3(24)

CLOSE

1(5)

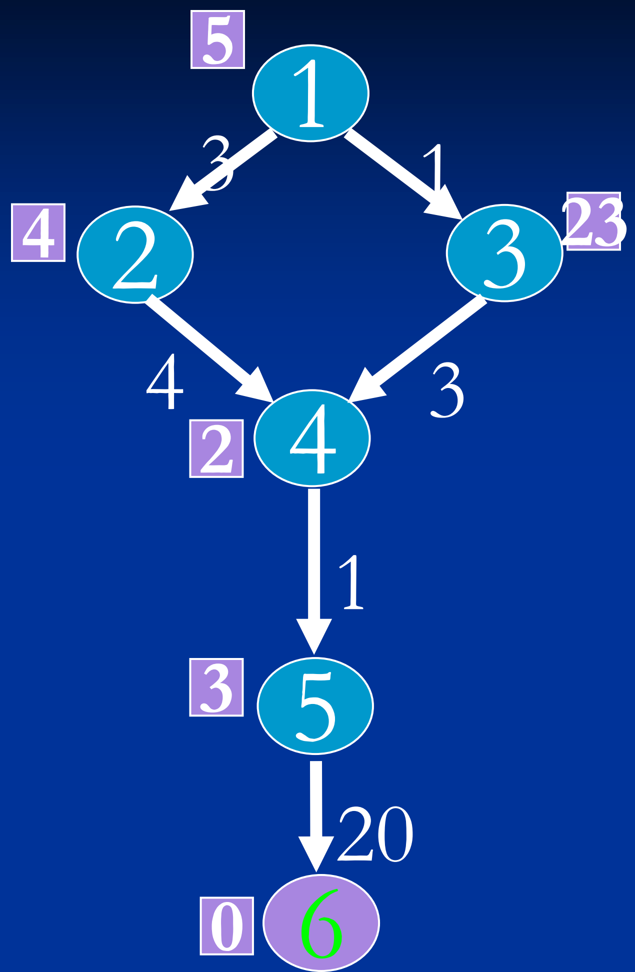


OPEN

3(24)

CLOSE

1(5) 2(7)

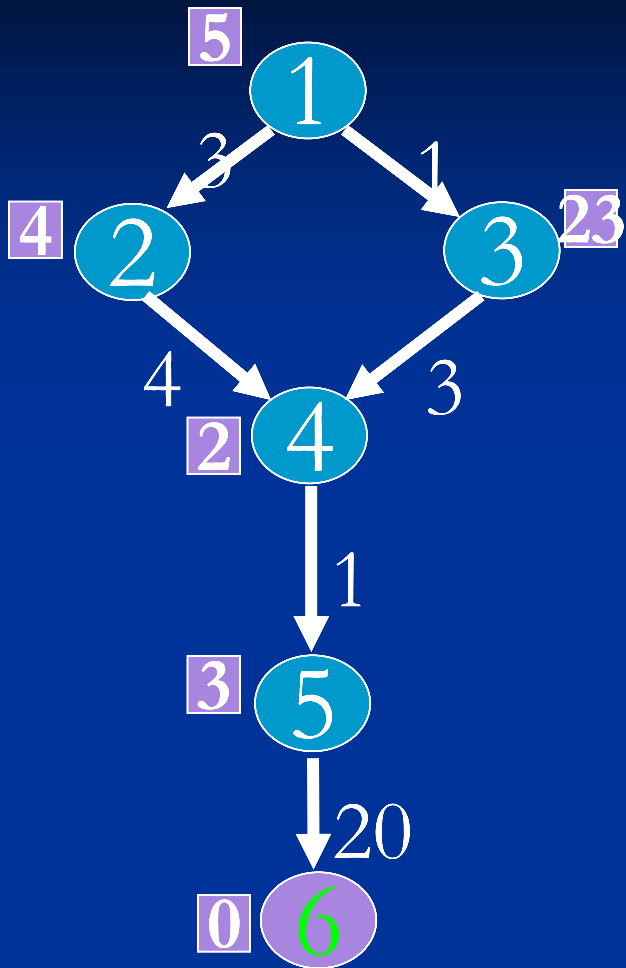


OPEN

3(24) 4(9)

CLOSE

1(5) 2(7)

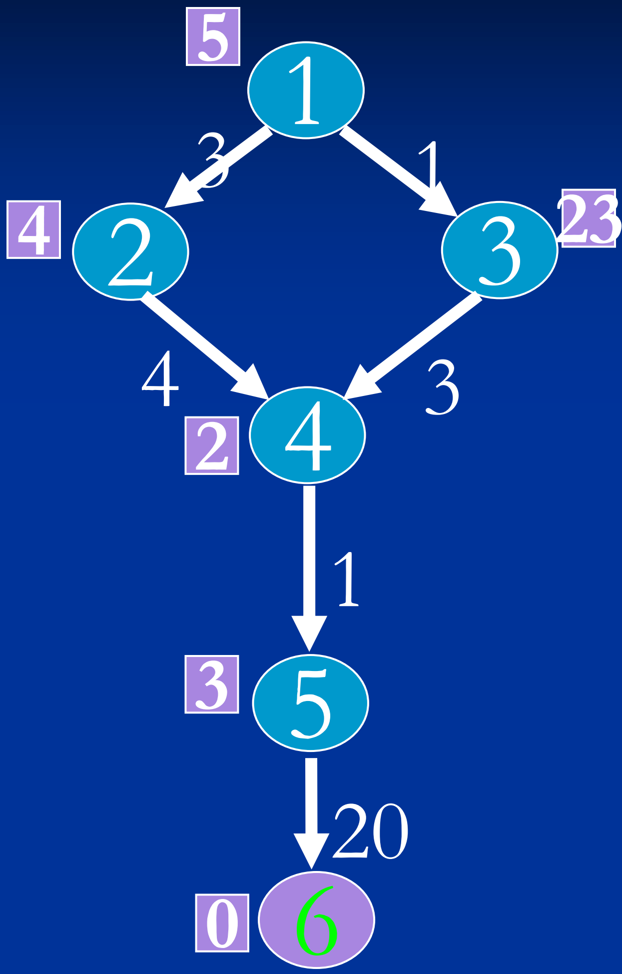


OPEN

3(24)

CLOSE

1(5) 2(7) 4(9)

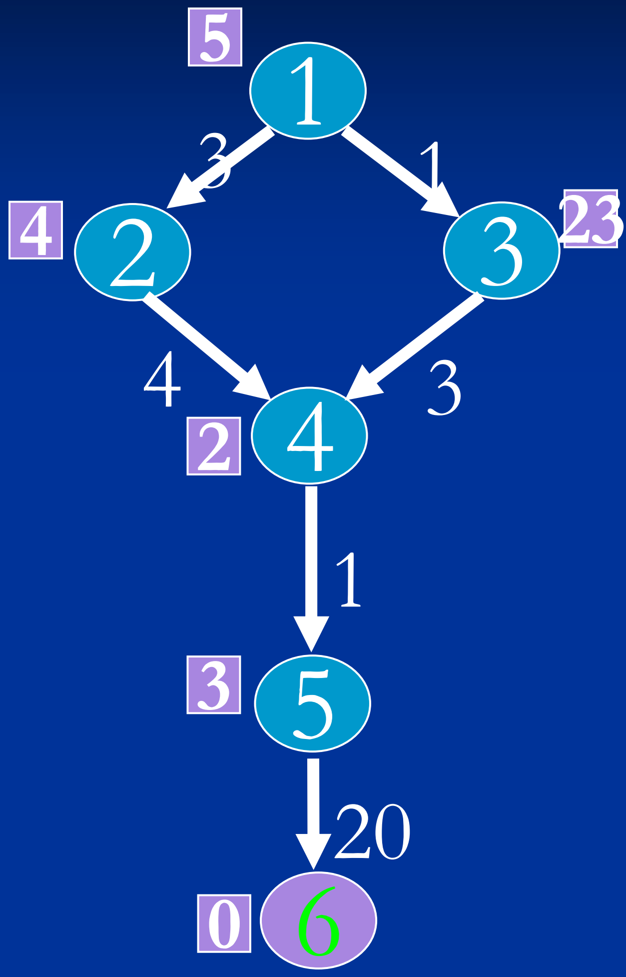


OPEN

3(24) 5(11)

CLOSE

1(5) 2(7) 4(9)

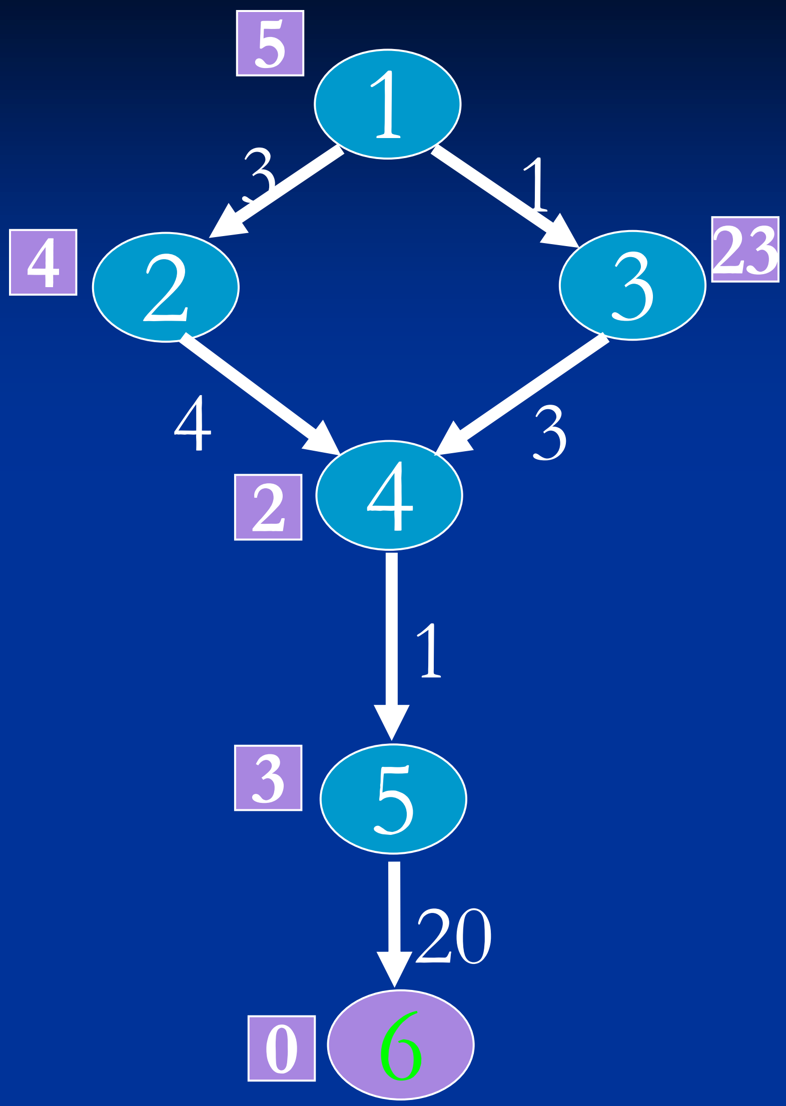


OPEN

3(24)

CLOSE

1(5) 2(7) 4(9) 5(11)

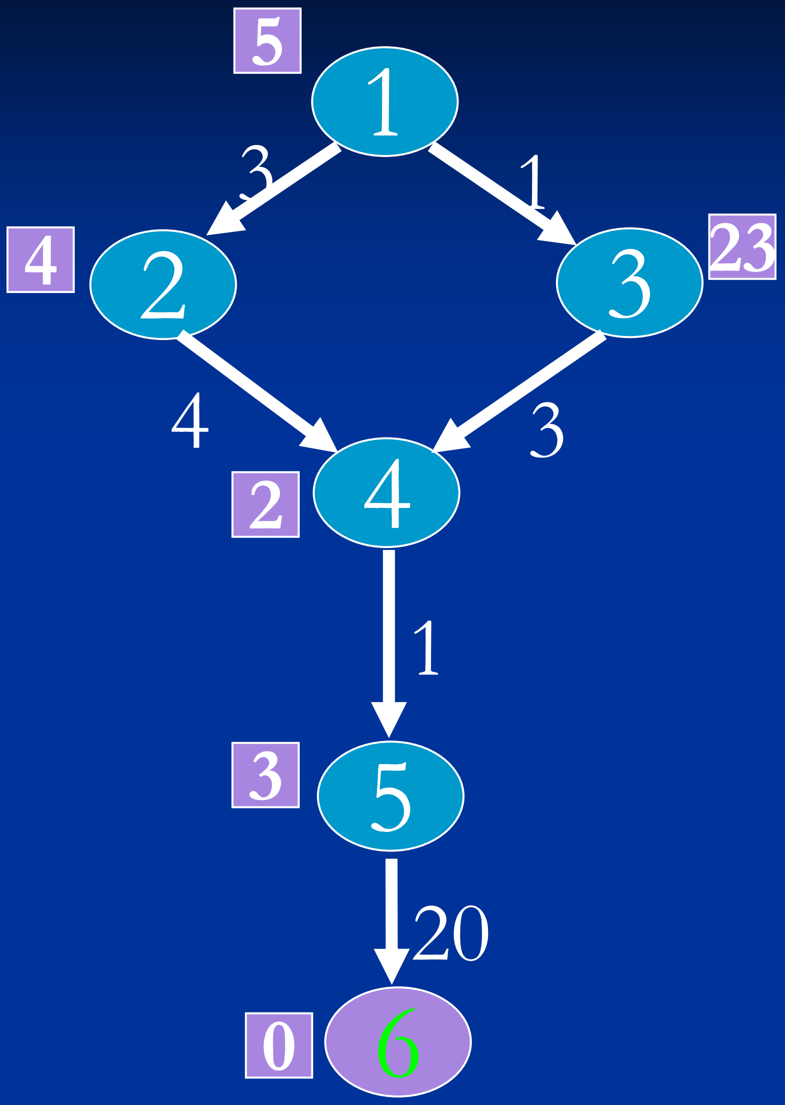


OPEN

3(24) 6(28)

CLOSE

1(5) 2(7) 4(9) 5(11)

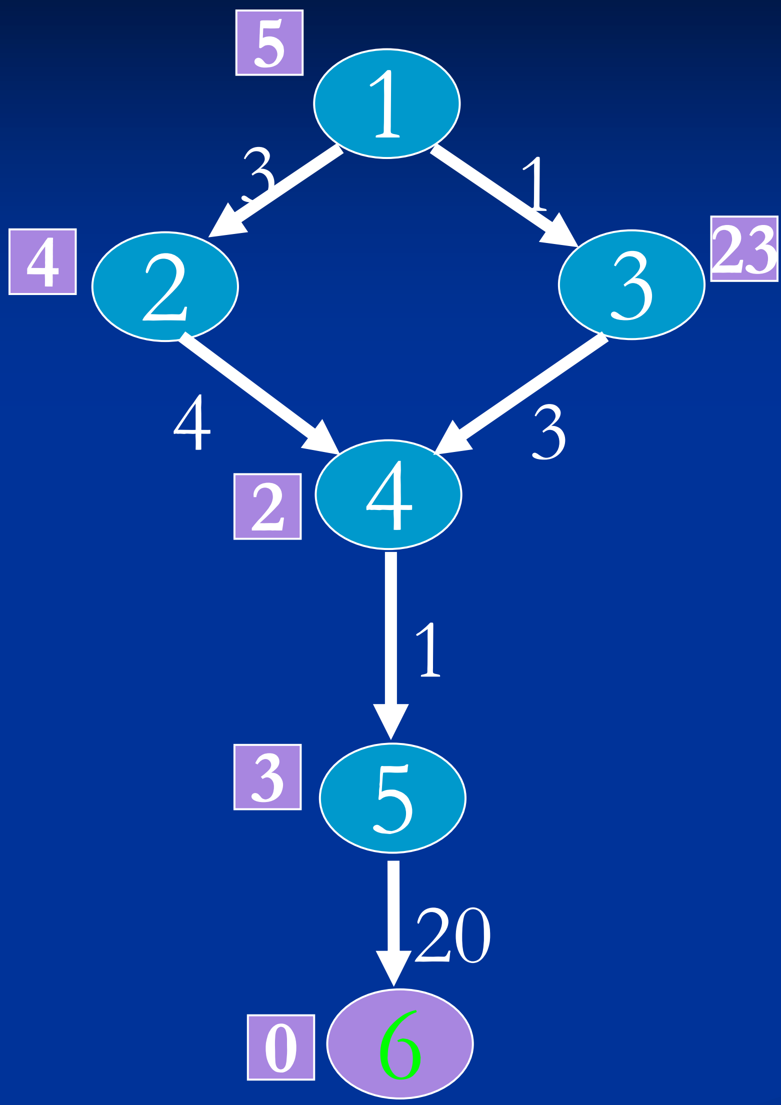


OPEN

6(28)

CLOSE

1(5) 2(7) 4(9) 5(11) 3(24)

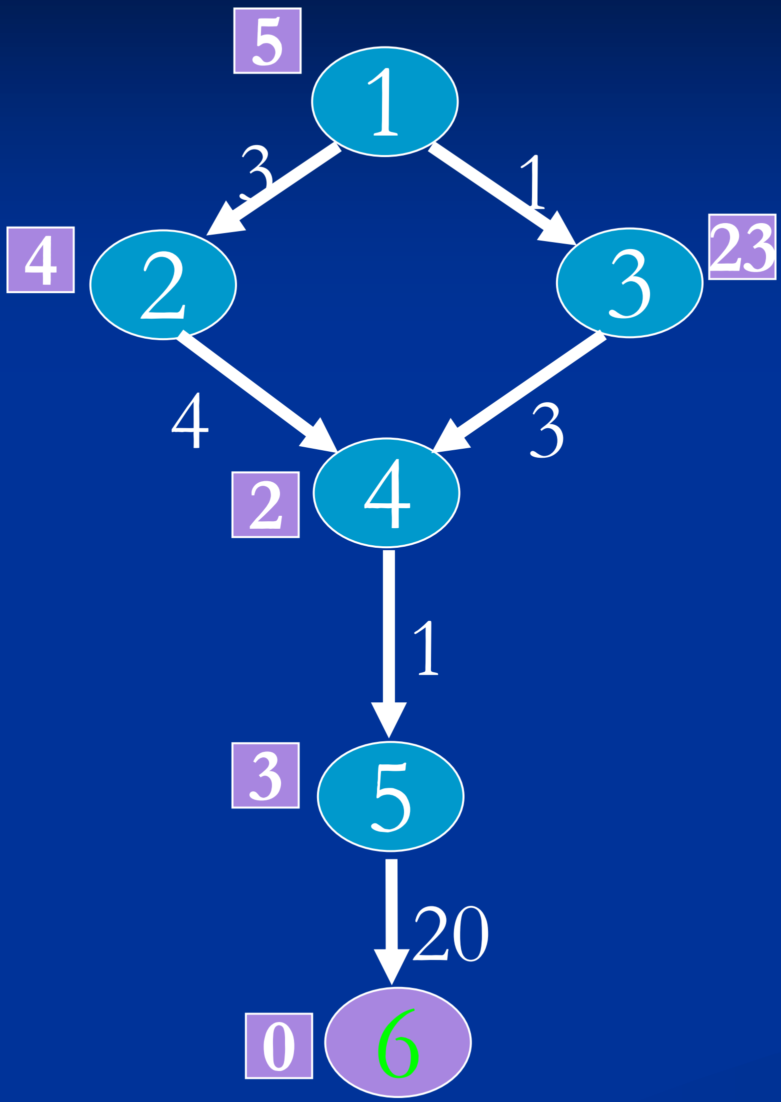


OPEN

6(28) 4(6)

CLOSE

1(5) 2(7) ~~4(9)~~ 5(11)
3(24)



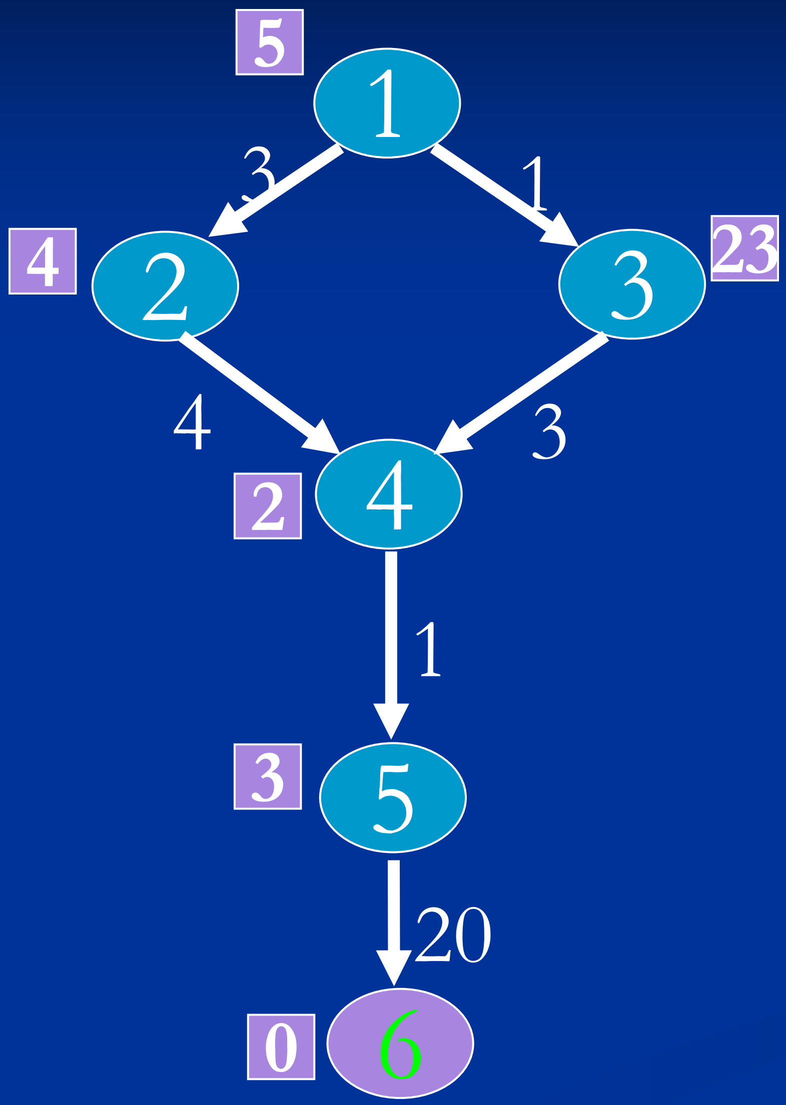
OPEN

6(28)

CLOSE

1(5) 2(7) 4(9) 5(11)

3(24) 4(6)



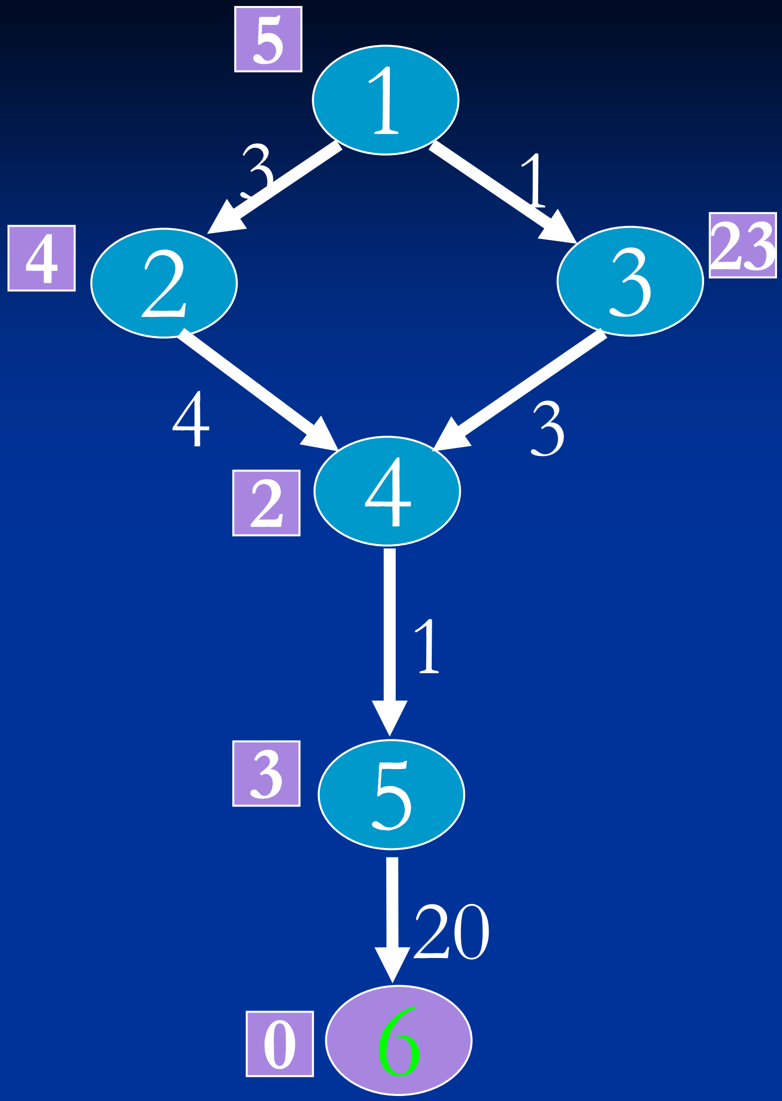
OPEN

6(28) 5(8)

CLOSE

1(5) 2(7) 4(9) ~~5(11)~~

3(24) 4(6)

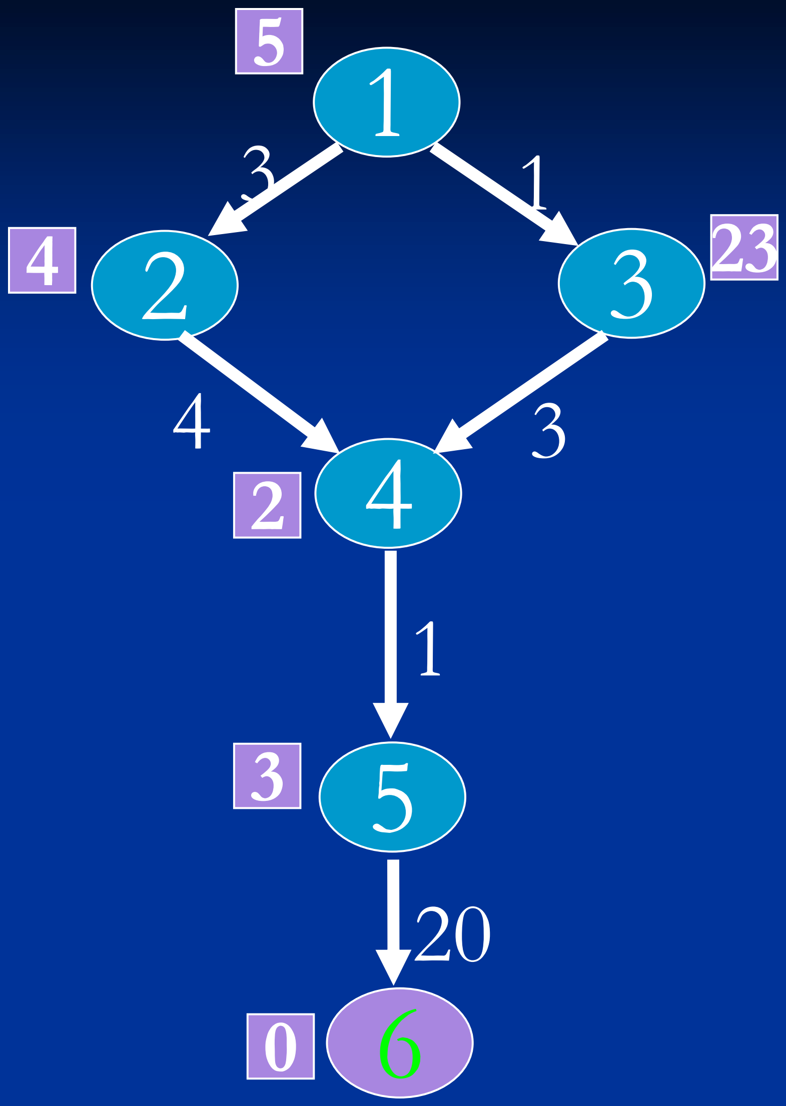


OPEN

6(28)

CLOSE

1(5) 2(7) 4(9) 5(11)
3(24) 4(6) 5(8)

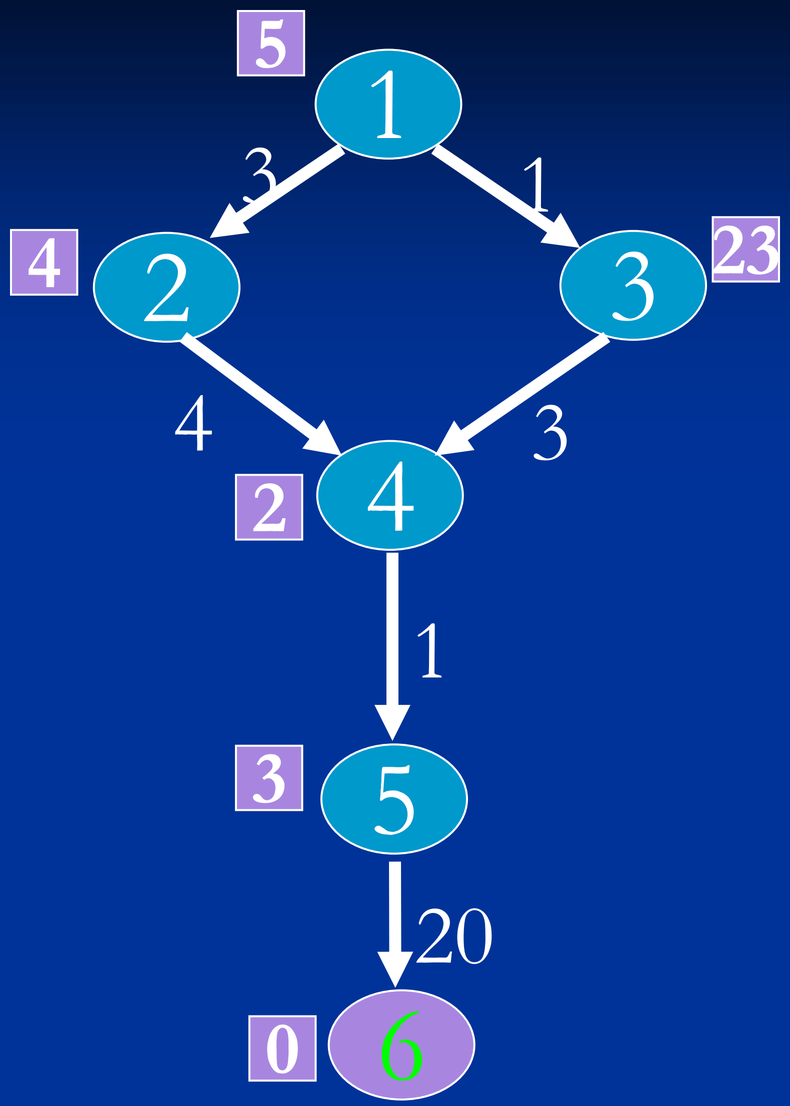


OPEN

~~6(28)~~ 6(25)

CLOSE

1(5) 2(7) 4(9) 5(11)
3(24) 4(6) 5(8)



OPEN

CLOSE

1(5) 2(7) 4(9) 5(11)

3(24) 4(6) 5(8) 6(25)

Results

- A heuristic is called admissible if it always under estimates, that is, we always have $h(n) \leq f^*(n)$, where $f^*(n)$ denotes the minimum distance to a goal state from state n
- For finite state spaces, A* always terminates
- Algorithm A* is admissible, that is, if there is a path from start state to a goal state, A* terminates by finding an optimal path

Results

- If A1 and A2 are two versions of A* such that A2 is more informed than A1 , then A1 expands at least as many states as does A2 (Because h2() is more informed then h1())
- If we are given two or more admissible heuristics for every state and we do not know which is more informed, then, we can take their max to get a stronger admissible heuristic at every state.

$$h(n) = \max (h1(n), h2(n))$$

A* Algorithm