Algorithmic Methods
Tricks of the Trade

5B

Algorithmic Techniques

Divide and Conquer

- In the military: strategy to gain or maintain power
- In computation:
  - Divide the problem into "simpler" versions of itself.
  - Conquer each problem using the same process (usually recursively).
  - Combine the results of the "simpler" versions to form your final solution.
- Example: Towers of Hanoi

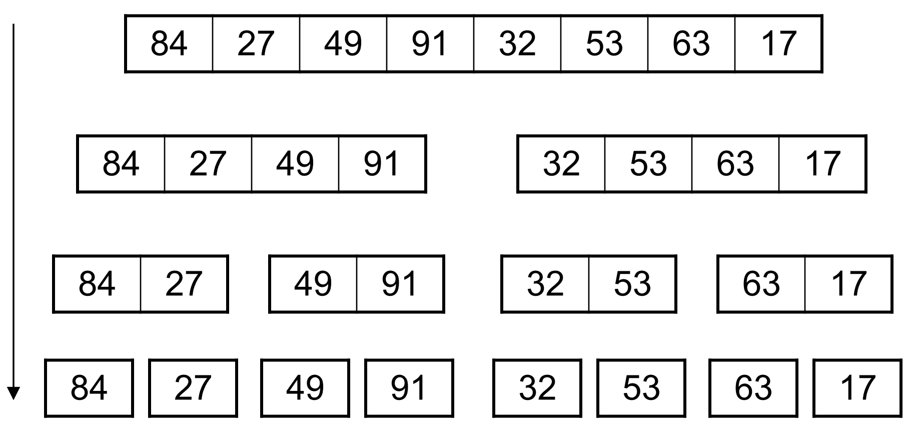
MergeSort

- Split the vector into 2 halves.*
- If the size of the half-vector is not 1,
  - Sort each half using merge sort.
- Set up a new vector.
- Merge step:

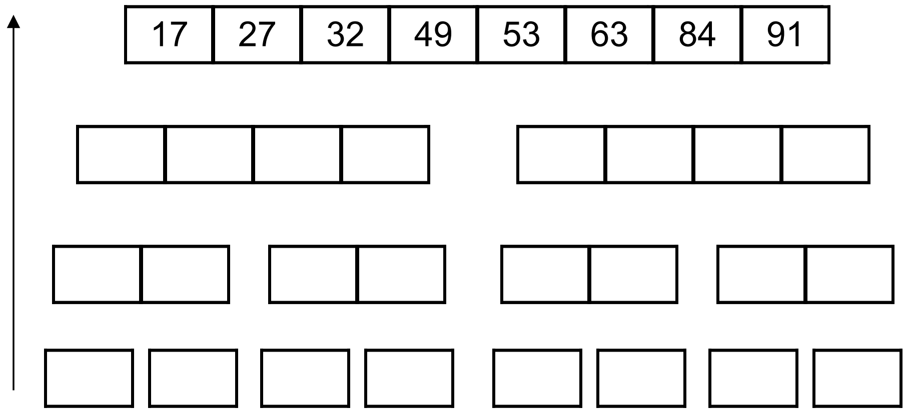
  Move the smallest of each half-vector into the new vector. Repeat this step until one of the half-vectors is empty. Then copy the remaining data in the non-empty half-vector into the new vector.

*If the vector has an odd number of elements, one vector will have one extra element.

MergeSort



MergeSort



Greedy Algorithms

- At each stage of the algorithm, the locally "best" step is taken.
- Goal: Develop the best solution using the locally best choices at each step.
- Example: Minimum spanning tree algorithm

  greedy!

  - Choose the lowest cost edge that connects to nodes already in the spanning tree without creating a cycle.

Huffman Coding
A Greedy Algorithm

- Compression technique to reduce information into the shortest optimal sequence of bits.
- Used in many computer applications:
  - ZIP – file compression
  - MP3 – music compression
  - JPEG – image compression

Huffman Coding

- Assume messages consist of the symbols A,B,C,D with the following average frequencies:
  - A occurs 45% of the time
  - B occurs 30% of the time
  - C occurs 20% of the time
  - D occurs 5% of the time
- Create a frequency table from highest to lowest frequency.

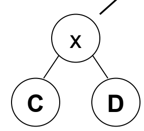
A	45
B	30
C	20
D	5

Building the Huffman Tree 1

A	45
B	30
C	20
D	5

greedy!

Remove two lowest frequency symbols and make a small tree with a new node.



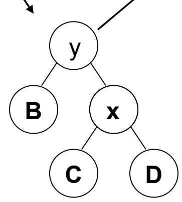
New node is reinserted with sum of the frequencies of its children.

A	45
B	30
x	25

Building the Huffman Tree 2

A	45
B	30
x	25

Remove two lowest frequency symbols and make a small tree with a new node.



New node is reinserted with sum of the frequencies of its children.

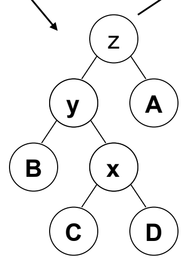
y	55
A	45

Building the Huffman Tree 3

y	55
A	45

z	100

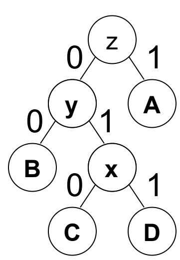
Remove two lowest frequency symbols and make a small tree with a new node.



New node is reinserted with sum of the frequencies of its children.

Labeling the Huffman Tree

Assign a 0 to each left branch.
Assign a 1 to each right branch.



A	1
B	00
C	010
D	011

Huffman codes are defined for each original letter by starting at the root of the tree and reading the 0s and 1s as you descend to the letter in a leaf.

Using the Huffman Tree

Encode the following message:
AABACDADC

A	1
B	00
C	010
D	011

Decode the following message:
01010011011

Using Fixed-Width Codes

ASCII (American Standard Code for Information Interchange)
Each character is represented as a 7-bit binary number.

A	1000001
B	1000010
C	1000011
D	1000100

Encode AABACDADC:

Using Fixed-Width Codes

What is the smallest fixed width code we can use for 4 characters?
2 bits yields
$2^2 = 4$ combinations
which is enough.

A	00
B	01
C	10
D	11

Encode AABACDADC:

Comparing Coding Schemes

When does Huffman Coding yield a shorter encoding?

When does Minimal Fixed Width Coding yield a shorter encoding?

Dijkstra's Shortest Path Algorithm

A Greedy Algorithm

- Find the shortest path through a connected weighted directed graph W of n nodes without cycles.
  - Notation: Let W[x,y] be the weight on the edge from node x to node y.
- Let SP[x] = the **shortest path** (as a function of the weights) from the starting node to node x.
- Let Pr[x] = the **predecessor** in the shortest path from the starting node to node x.
- Initially SP[x] = ∞ for all nodes, except the starting node *s*, where SP[*s*] = 0.

Dijkstra's Shortest Path Algorithm

1. Let X be the set of nodes where the shortest path has been completely determined. (Initially X is empty.)
2. While all nodes are not in set X, do the following:
   a. Extract the node u that is not in X that has the smallest SP value and add u to the set X.

greedy!

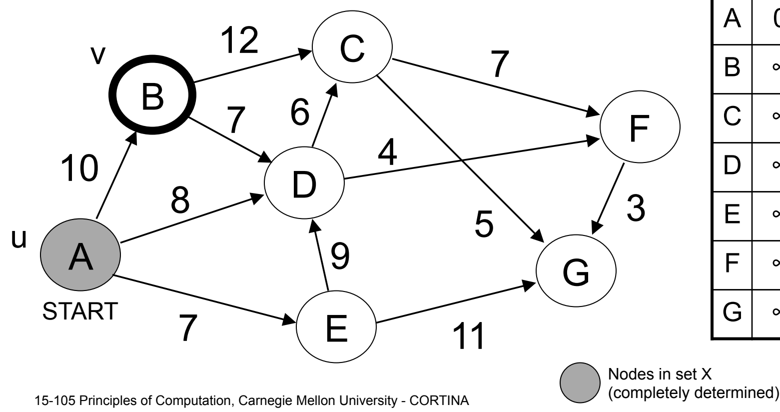
   b. For each node v adjacent to u that is not in X, do the following:
      i. if SP[v] > SP[u] + W[u,v], then set SP[v] equal to SP[u] + W[u,v] and set Pr[v] equal to u.

"relaxing the edge"

Shortest Path Example

Examine each neighbor of A: Is SP[v] > SP[u] + W[u,v] ?

SP[B] > SP[A] + W[A,B]
∞ > 0 + 10 YES... Set SP[B] to 0+10=10 and Pr[B]=A

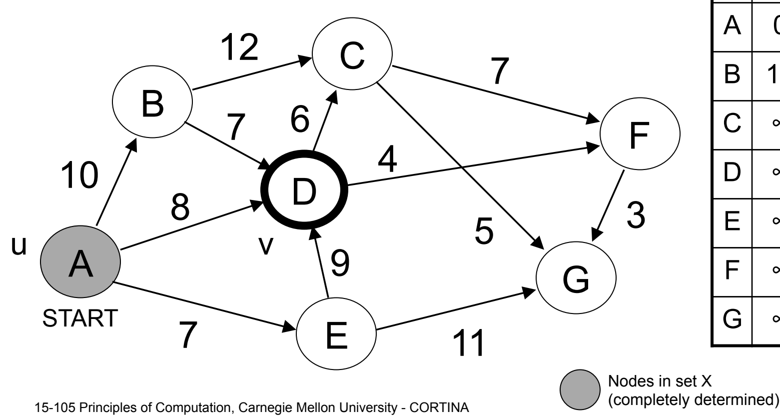


x	SP	Pr
A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	

Shortest Path Example

Examine each neighbor of A: Is SP[v] > SP[u] + W[u,v] ?

SP[D] > SP[A] + W[A,D]
∞ > 0 + 8 YES... Set SP[D] to 0+8=8 and Pr[D]=A

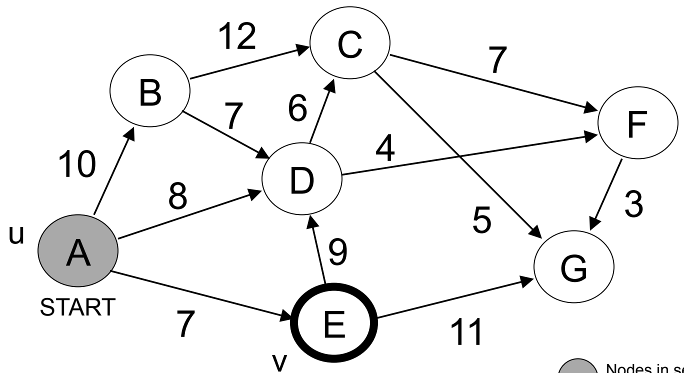


x	SP	Pr
A	0	
B	10	A
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	

Shortest Path Example

Examine each neighbor of A: Is SP[v] > SP[u] + W[u,v] ?

SP[E] > SP[A] + W[A,E]
∞ > 0 + 7 YES... Set SP[E] to 0+7=7 and Pr[E]=A

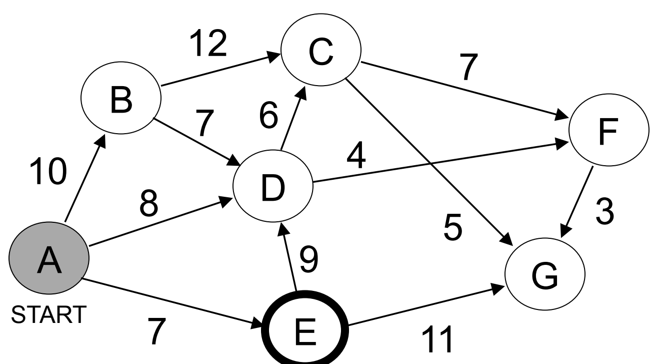


x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	∞	
F	∞	
G	∞	

Nodes in set X (completely determined)

Shortest Path Example

Select the node with the smallest SP value that is not in set X (shaded).



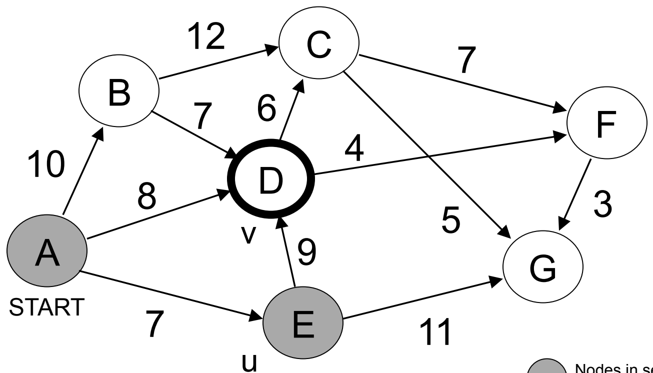
x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	7	A
F	∞	
G	∞	

Nodes in set X (completely determined)

Shortest Path Example

Examine each neighbor of E: Is SP[v] > SP[u] + W[u,v] ?

SP[D] > SP[E] + W[E,D]
8 > 7 + 9   NO



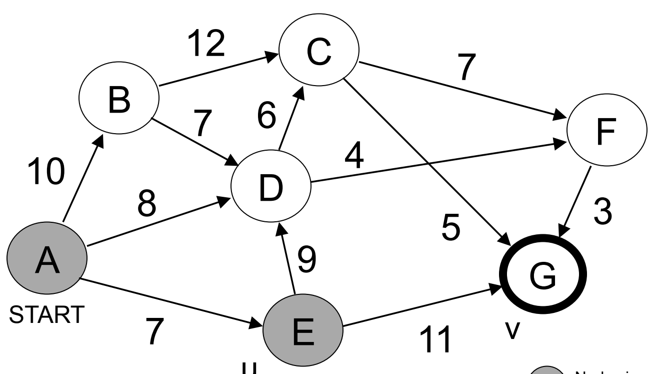
x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	7	A
F	∞	
G	∞	

Nodes in set X (completely determined)

Shortest Path Example

Examine each neighbor of E: Is SP[v] > SP[u] + W[u,v] ?

SP[G] > SP[E] + W[E,G]
∞ > 7 + 11   YES... Set SP[G] to 7+11=18 and Pr[G]=E

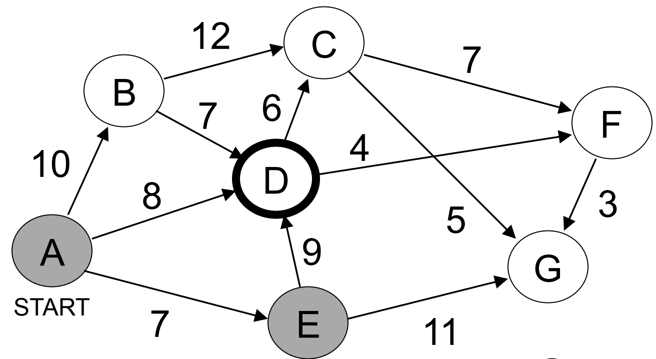


x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	7	A
F	∞	
G	∞	

Nodes in set X (completely determined)

Shortest Path Example

Select the node with the smallest SP value
that is not in set X (shaded).



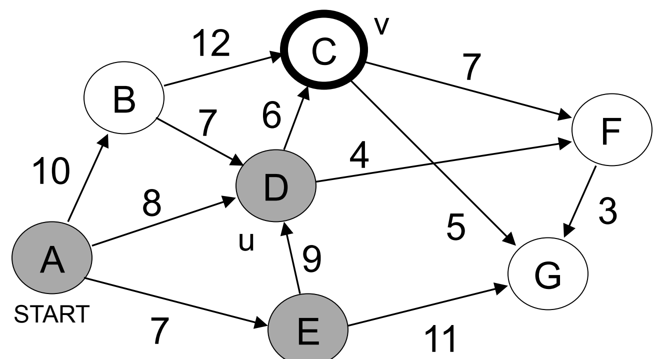
x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	7	A
F	∞	
G	18	E

Nodes in set X
(completely determined)

Shortest Path Example

Examine each neighbor of D: Is SP[v] > SP[u] + W[u,v] ?

SP[C] > SP[D] + W[D,C]
∞ > 8 + 6 YES... Set SP[C] to 8+6=14 and Pr[C]=D



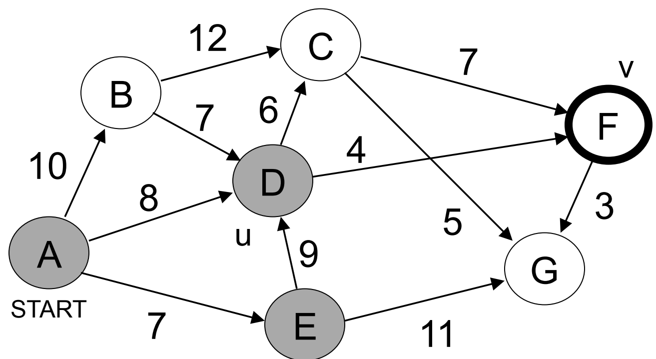
x	SP	Pr
A	0	
B	10	A
C	∞	
D	8	A
E	7	A
F	∞	
G	18	E

Nodes in set X
(completely determined)

Shortest Path Example

Examine each neighbor of D: Is SP[v] > SP[u] + W[u,v] ?

SP[F] > SP[D] + W[D,F]
∞ > 8 + 4   YES... Set SP[F] to 8+4=12 and Pr[F]=D

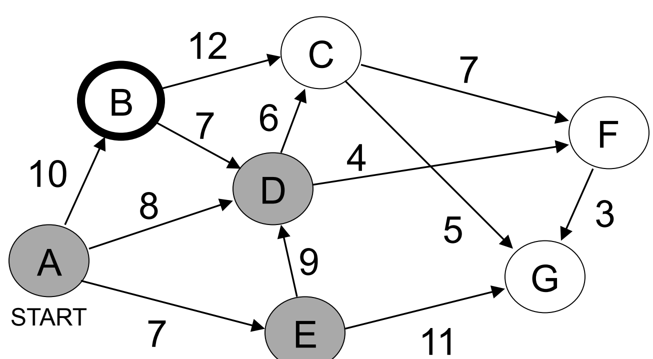


x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	∞	
G	18	E

Nodes in set X
(completely determined)

Shortest Path Example

Select the node with the smallest SP value
that is not in set X (shaded).



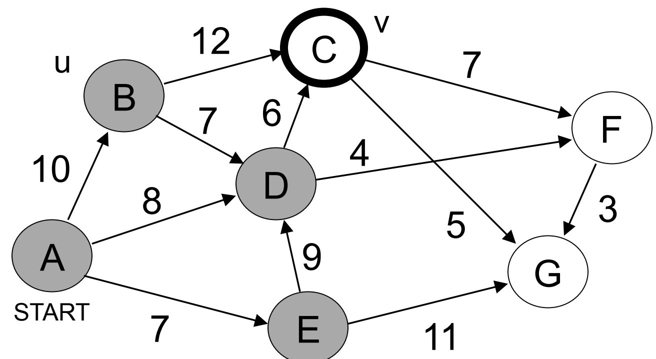
x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	18	E

Nodes in set X
(completely determined)

Shortest Path Example

Examine each neighbor of B: Is SP[v] > SP[u] + W[u,v] ?

SP[C] > SP[B] + W[B,C]
14 > 10 + 12 NO

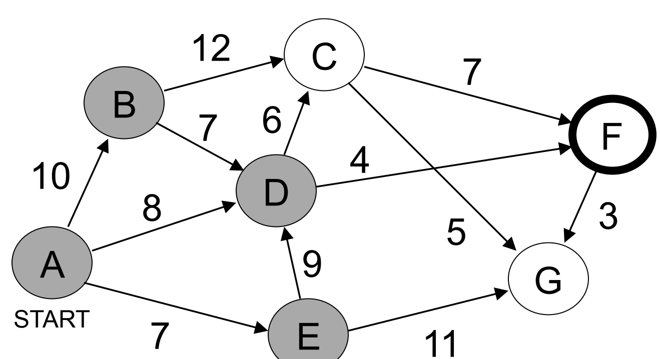


x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	18	E

Nodes in set X (completely determined)

Shortest Path Example

Select the node with the smallest SP value that is not in set X (shaded).



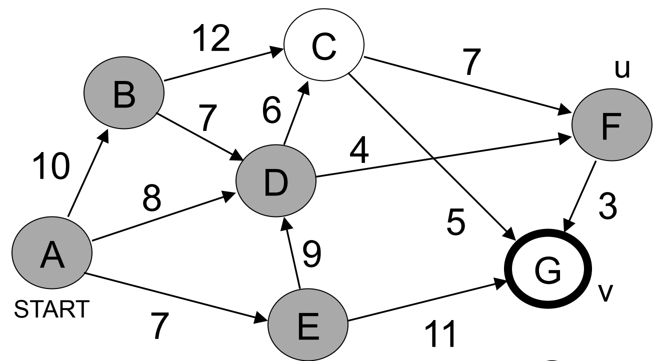
x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	18	E

Nodes in set X (completely determined)

Shortest Path Example

Examine each neighbor of F: Is SP[v] > SP[u] + W[u,v] ?

SP[G] > SP[F] + W[F,G]
18 > 12 + 3 YES... Set SP[G] to 12+3=15 and Pr[G]=F

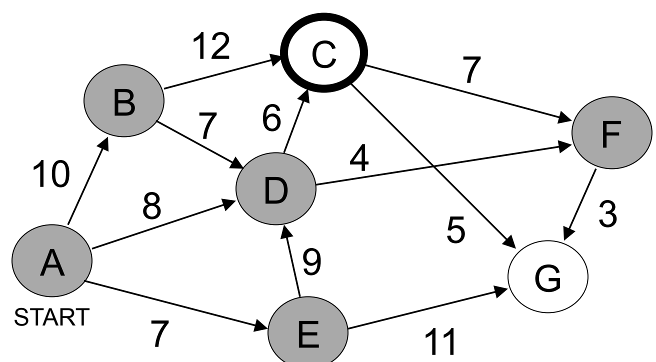


x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	18	E

Nodes in set X (completely determined)

Shortest Path Example

Select the node with the smallest SP value that is not in set X (shaded).



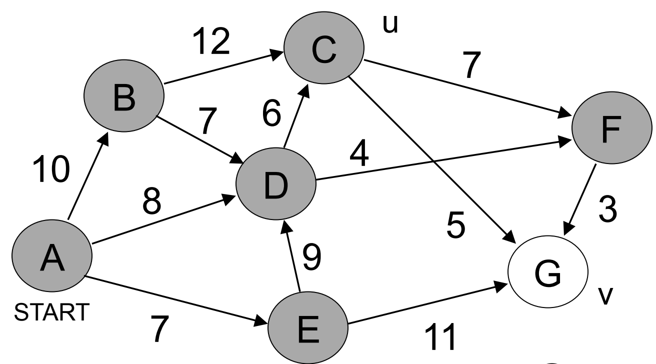
x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	15	F

Nodes in set X (completely determined)

Shortest Path Example

Examine each neighbor of C: Is SP[v] > SP[u] + W[u,v] ?

SP[G] > SP[C] + W[C,G]
15 > 14 + 5 NO

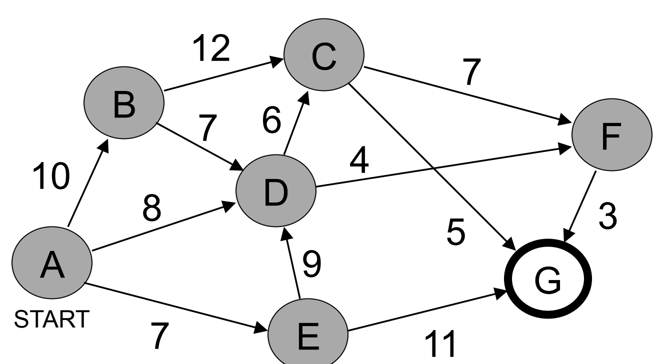


x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	15	F

Nodes in set X (completely determined)

Shortest Path Example

Select the node with the smallest SP value that is not in set X (shaded).
DONE - Last node chosen.

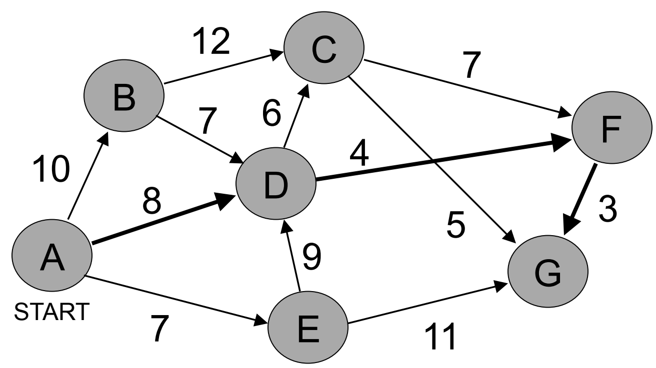


x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	15	F

Nodes in set X (completely determined)

Shortest Path Example

Shortest path from A to G costs 15.
The shortest path is A to D to F to G.
(Start with G and use Pr values to work backwards.)



x	SP	Pr
A	0	
B	10	A
C	14	D
D	8	A
E	7	A
F	12	D
G	15	F

Other Algorithmic Techniques

- Probabilistic
  - An algorithm makes some random choices to guide its computation to guarantee good performance most of the time.
- Genetic
  - An algorithm that uses ideas from evolutionary biology to generate solutions that mutate, merge and die out in successive generations until an optimal answer is found.
- Heuristic
  - An algorithm that finds a solution that is nearly optimal in much faster time than trying to find the absolute best answer in much longer time. (More on this in A.I.)