University of Nottingham
UK | CHINA | MALAYSIA

COMP2054 Tutorial Session 7: DP, Change Giving and MSTs

Rebecca Tickle
Warren Jackson
AbdulHakim Ibrahim

Session outcomes

- Understand how dynamic programming can be used to solve (min-coin) change giving problem.
- Know what an MST is.
- Calculate the MST from a graph.

University of Nottingham
UK | CHINA | MALAYSIA

Dynamic Programming

(Min-Coins) Change Giving

Dynamic Programming

A technique used to solve a larger problem by solving smaller "decomposed" subproblems and building-up to the original problem.

Example: Subset Sum (decision problem) – given a multiset of integers, x[0] … x[n-1], is there a subset that sums to a given value, K?

- Smallest subproblem has a multiset of cardinality zero.
- Build up to the original problem by adding the next element from the multiset until all elements are added (or the target value is confirmed).

Subset Sum Example

Example: Subset Sum (decision problem) – given a multiset of integers, x[0] … x[n-1], is there a subset that sums to a given value, K?

- Smallest subproblem has a multiset of cardinality zero.
- Build up to the original problem by adding the next element from the multiset until all elements are added (or the target value is confirmed).

Given $X = \{1,2,2,5\}$ and $K = 4$:

Value/index	0	1	2	3	4
Possible?	0	0	0	0	0

Subset Sum Example

Example: Subset Sum (decision problem) – given a multiset of integers, x[0] … x[n-1], is there a subset that sums to a given value, K?

- Smallest subproblem has a multiset of cardinality zero.
- Build up to the original problem by adding the next element from the multiset until all elements are added (or the target value is confirmed).

Given $X = \{1,2,2,5\}$ and $K = 4$: **Considering {}**

Value/index	0	1	2	3	4
Possible?	0	0	0	0	0

Value/index	0	1	2	3	4
Possible?	1	0	0	0	0

Subset Sum Example

Given $X = \{1,2,2,5\}$ and $K = 4$:

Value/index	0	1	2	3	4
Possible?	1	0	0	0	0

Considering {1}

Value/index	0	1	2	3	4
Possible?	1	1	0	0	0

Subset Sum Example

Given $X = \{1,2,2,5\}$ and $K = 4$:

Value/index	0	1	2	3	4
Possible?	1	0	0	0	0

Value/index	0	1	2	3	4
Possible?	1	1	0	0	0

Considering {1,2}

Value/index	0	1	2	3	4
Possible?	1	1	1	1	0

Subset Sum Example

Given $X = \{1,2,2,5\}$ and $K = 4$:

Value/index	0	1	2	3	4
Possible?	1	0	0	0	0

Value/index	0	1	2	3	4
Possible?	1	1	0	0	0

Value/index	0	1	2	3	4
Possible?	1	1	1	1	0

Considering {1,2,2}

Value/index	0	1	2	3	4
Possible?	1	1	1	1	0

2 + x[2] == 4; return success;

DP for Min-Coins-Change-Giving

Given a multiset of coins, S, and a target value, K, find the subset of S which sums to K **with the smallest cardinality**.

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

Given a multiset of coins, S, and a target value, K, find the subset of S which sums to K **with the smallest cardinality**.

Basic idea:

- For each element in S.
- Scan backwards over an array Y which tells us whether each value was possible using some number of coins for the subset already considered.
- Introducing a coin of value $x[i]$ means we can give $m + x[i]$ change using $\min(Y[m + x[i]], Y[m] + 1)$ coins.

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

- Given $S = \{\mathbf{2}, 3,5\}$ and $K = 5$:
- Initialise $Y = [0, -1, -1, -1, -1]$
- First iteration ($\boldsymbol{i = 0}$):

Do nothing for m=3 to m=1 as Y[m]<0

0	1	2	3	4	5
0	-1	-1	-1	-1	-1

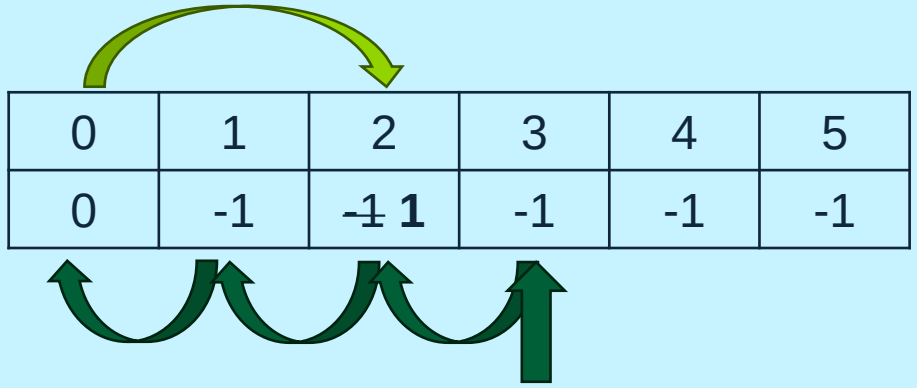
m=3

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

- Given $S = \{\mathbf{2}, 3,5\}$ and $K = 5$:
- Initialise $Y = [0, -1, -1, -1, -1]$
- First iteration ($\boldsymbol{i = 0}$):

Coins for sum “0+2” = 0+1 = 1



0	1	2	3	4	5
0	-1	~~-1~~ **1**	-1	-1	-1

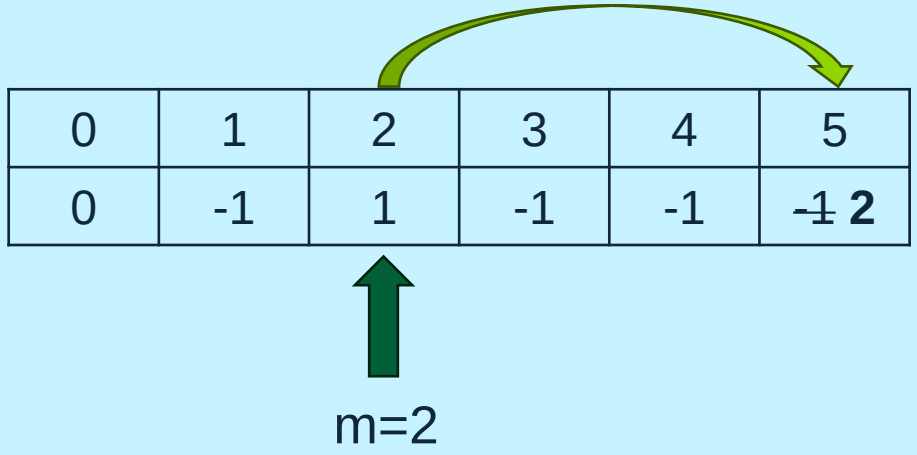
m=0

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

- Given $S = \{2, \mathbf{3}, 5\}$ and $K = 5$:
- Second iteration ($\boldsymbol{i = 1}$):

Coins for sum “2+3” = 1+1 = 2

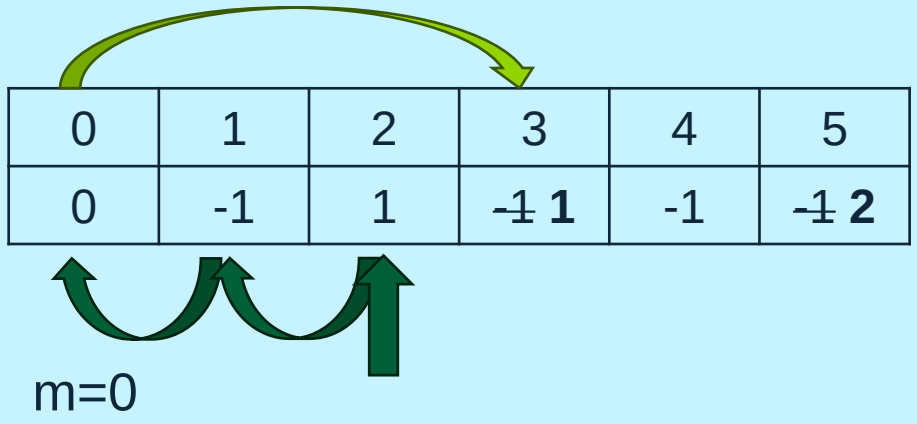



```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

- Given $S = \{2, \mathbf{3}, 5\}$ and $K = 5$:
- Second iteration ($\boldsymbol{i = 1}$):

Coins for sum “0+3” = 0+1 = 1



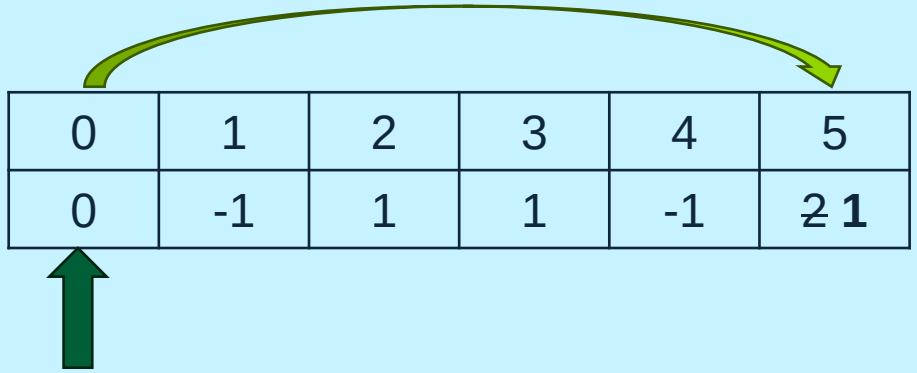
We found that we **can** make change of value 5 using 2 coins, but can we stop here? (Min coins)

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

DP for Min-Coins-Change-Giving

- Given $S = \{2,3,\mathbf{5}\}$ and $K = 5$:
- Second iteration ($\boldsymbol{i = 2}$):

Coins for sum “0+5”
= min(Y[5], Y[0] + 1)
= min(2, 1)
= 1



0	1	2	3	4	5
0	-1	1	1	-1	~~2~~ **1**

m=0

```
input: x[0], …, x[n-1] and K
init: Y[0] = 0, Y[m] = -1 (∀m > 0)
for(i=0; i<n; i++) {
  for(m=K-x[i]; m>=0; m--) {
    if(Y[m] >= 0) {
      if(Y[m+x[i]] == -1) {
        Y[m+x[i]] = Y[m]+1
      } else {
        Y[m+x[i]] = min(Y[m+x[i]], Y[m]+1)
      }
    }
  }
}
```

Questions

Q1. Given the set of coins $\{1,1,3,5,9\}$, what is the minimum number of coins required to give change of 10?

Q2. Given the set of coins $\{2,2,2,5\}$, what is the minimum number of coins required to give change of 6?

Q3. Does the DP algorithm still work if we scan the array, Y, forwards instead of backwards? If yes, explain; if no, give a counter example.

Answers

Q1. Given the set of coins {1,1,3,5,9}, what is the minimum number of coins required to give change of 10?

K = 10 can be achieved with 2 coins.

Initialise Y:

0	1	2	3	4	5	6	7	8	9	10
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

i=0, x[i]=1:

0	1	2	3	4	5	6	7	8	9	10
0	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1

i=1, x[i]=1:

0	1	2	3	4	5	6	7	8	9	10
0	**1**	**2**	-1	-1	-1	-1	-1	-1	-1	-1

i=2, x[i]=3:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	**1**	**2**	**3**	-1	-1	-1	-1	-1

i=3, x[i]=5:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	1	2	**1**	**2**	**3**	**2**	**3**	**4**

i=4, x[i]=9:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	1	2	1	2	3	2	**1**	**2**

Answers

Q2. Given the set of coins {2,2,2,5}, what is the minimum number of coins required to give change of 6?

Initialise Y:

0	1	2	3	4	5	6
0	-1	-1	-1	-1	-1	-1

i=0, x[i]=2:

0	1	2	3	4	5	6
0	-1	**1**	-1	-1	-1	-1

i=1, x[i]=2:

0	1	2	3	4	5	6
0	-1	**1**	-1	**2**	-1	-1

i=2, x[i]=2:

0	1	2	3	4	5	6
0	-1	**1**	-1	**2**	-1	**3**

i=3, x[i]=5:

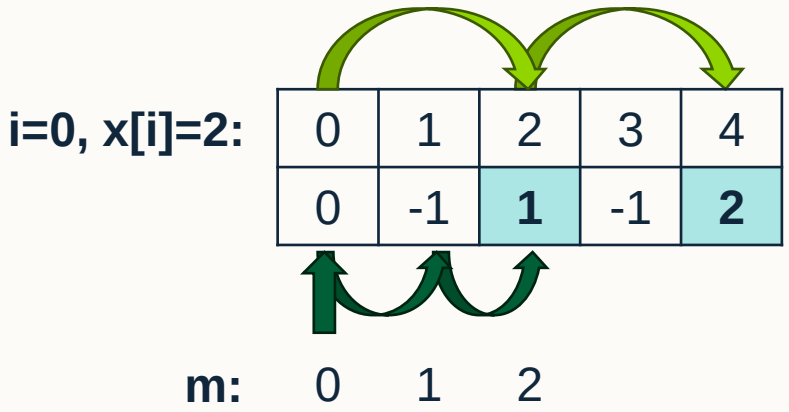
0	1	2	3	4	5	6
0	-1	1	-1	2	**1**	3

K = 6 can be achieved with 3 coins.

Answers

Q3. Does the DP algorithm still work if we scan the array, Y, forwards instead of backwards? If yes, explain; if no, give a counter example.

Using the inputs: {2,1,1} and K=4. If we scan Y forwards (i.e. we start the loop variable m at 0 and increment it at every step until we reach K-x[i]), then we would get the following first iteration:



We should only be considering a single coin in the first iteration, but Y has been updated to supposedly show that 4 is achievable with two coins. However, we can see from our input coins {2,1,1} that we should only be able to achieve a sum of 4 **by using all coins.**

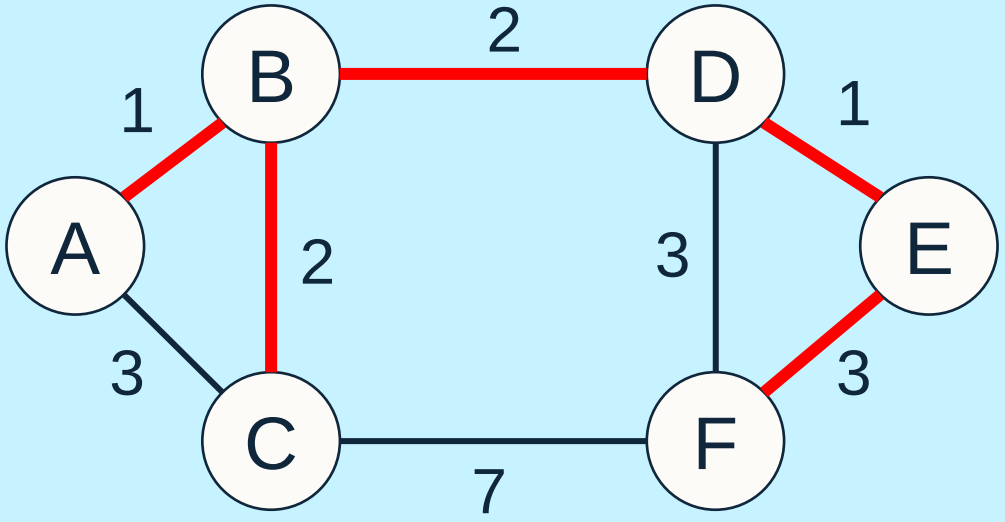
Consequently, the DP algorithm is not guaranteed to give the correct answer for all problems if we can Y forwards.

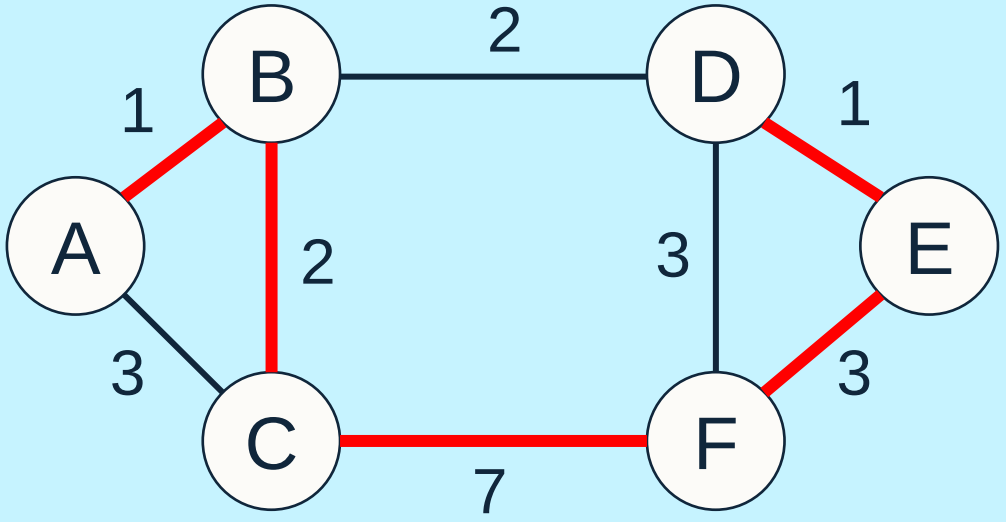
University of Nottingham
UK | CHINA | MALAYSIA

Minimum Spanning Trees

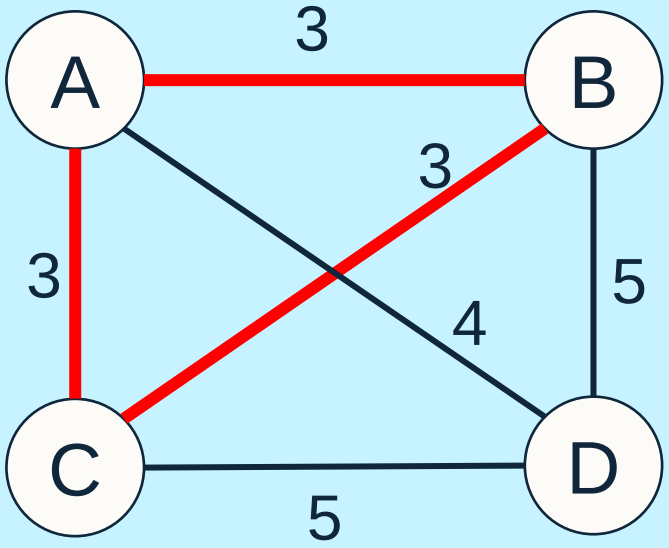
Prim’s Algorithm: An optimal greedy approach

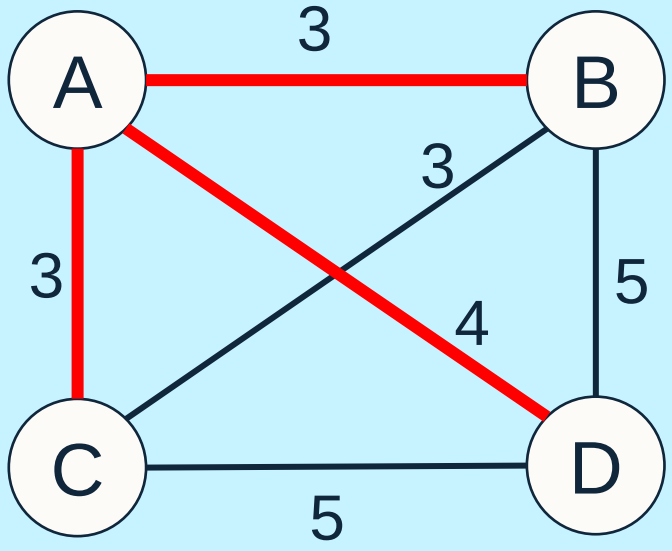
Which of the following is an MST?





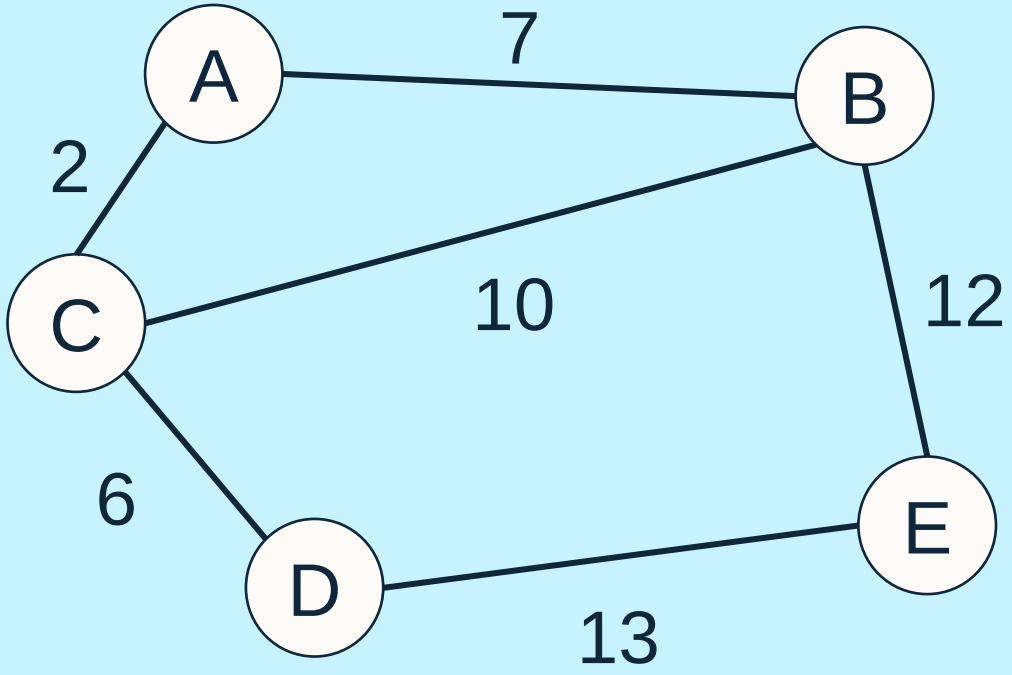
Which of the following is an MST?





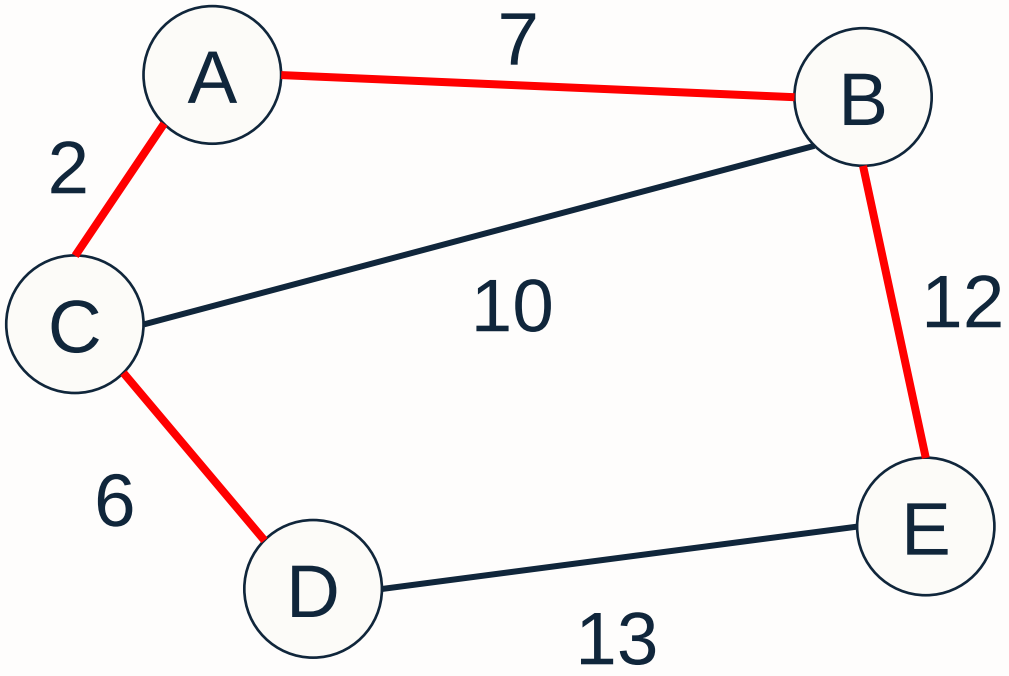
MST: Prim's Algorithm

Given the below graph and starting node A, create an associated minimum spanning tree. You should show all steps and give the final total cost of the MST.



MST: Prim's Algorithm

Given the below graph and starting node A, create an associated minimum spanning tree. You should show all steps and give the final total cost of the MST.

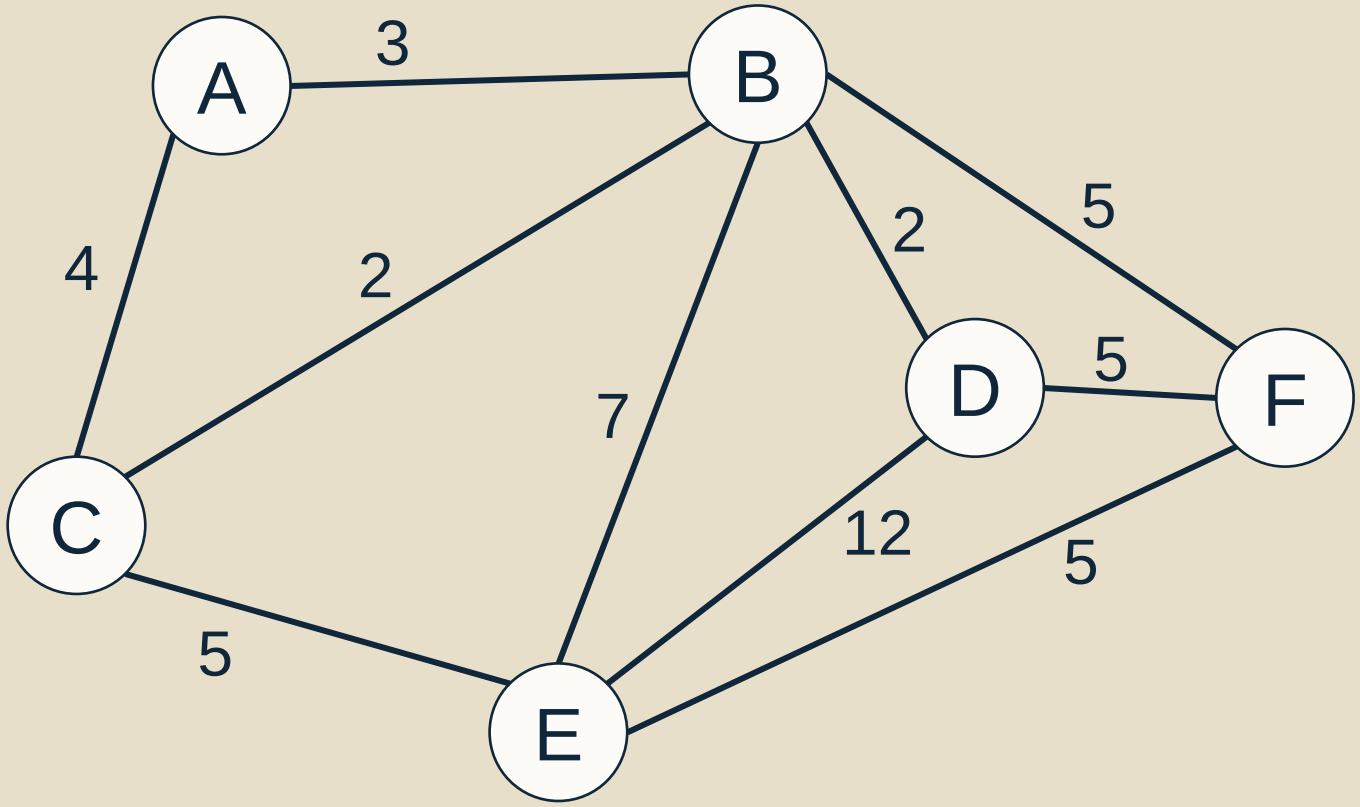


MST	Nodes not in MST
Start at node A	{B,C,D,E}
{(A,C)}	{B,D,E}
{(A,C), (C,D)}	{B,E}
{(A,C), (C,D), (A,B)}	{E}
{(A,C), (C,D), (A,B), (B,E)}	{}

$$\text{Total cost} = 2 + 6 + 7 + 12 = 27$$

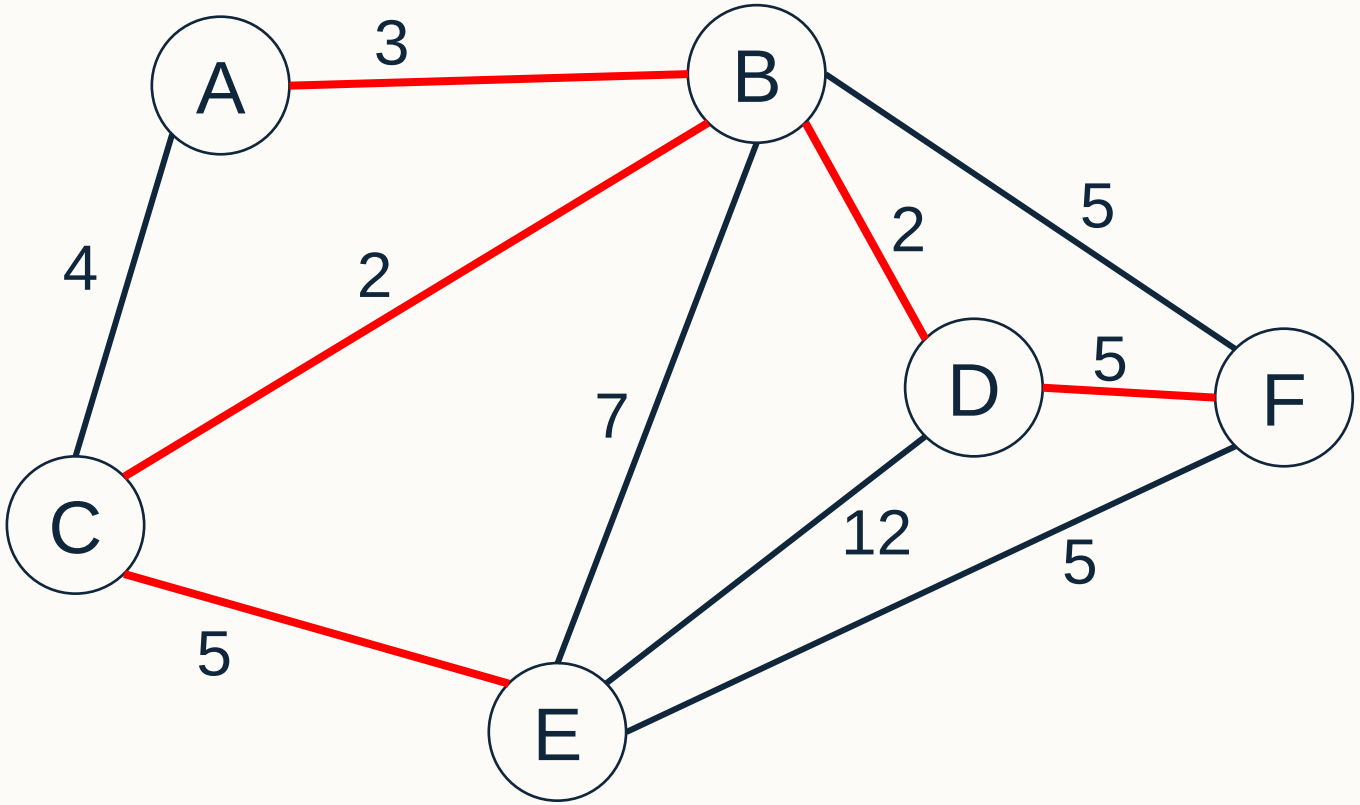
MST Question

Given the below graph and **starting node D**, create an associated minimum spanning tree. You should show all steps and give the final total cost of the MST.



Answer

Given the below graph and **starting node D**, create an associated minimum spanning tree. You should show all steps and give the final total cost of the MST.



MST	Nodes not in MST
Start at node D	{A,B,C,E,F}
{(D,B)}	{A,C,E,F}
{(D,B), (B,C)}	{A,E,F}
{(D,B), (B,C), (B,A)}	{E,F}
{(D,B), (B,C), (B,A), (C,E)}	{F}
{(D,B), (B,C), (B,A), (C,E), (D,F)}	{}

$$\text{Total cost} = 2 + 2 + 3 + 5 + 5 = 17$$

Note: there are multiple possible MSTs for this question. If you have two candidate edges with the same cost, you can randomly pick one.

University of Nottingham
UK | CHINA | MALAYSIA

Thank you

Next week – Floyd-Warshall Algorithm