Problem 1

a) The optimal substructure of the solution would be $A[i+1:n]$. if n was the total number of items, and i started at 0.

b)

```
item => object that contains a weight and value
arr => array containing items

def sort(arr):
   #sorts arr in descending order by the items value to weight
   ratio, so by item.value/item.weight

def fractionalKnapsack(W, arr):
   result = 0
   sort(arr)# sorts in descending order

   for item in arr:
      if item.weight <= W:
         result = result + item.value
         W = W - item.weight
      else:
         result = result + ((item.value/item.weight)*W)
         #stuff as much of the last item possible

   return result
```

c) The greedy algorithm works, because we steal the maximum valued item per weight. So we are maximizing the value in our knapsack for each weight value.

Problem 2

a)

	A	B	C	D	E
Entry Time	0	1	4	5	2
Finish Time	7	3	6	5	2

b)

1. First, we have an array that contains all the vertices and their completion status (unvisited, in progress, all done).
2. We pick and random vertex to start at. and run DFS. When DFS is finished we store the current time in $curr\_time$
3. We then look through the array and pick the firs vertex that is unvisited and run DFS with $curr\_time$ instead of 0
4. Repeat step 3 until all vertices are labelled as 'all done'

The total runtime of this algorithm should be $O(n+m)$

Problem 3

a) b)

iteration	A	B	C	D	E	F	G	H
0	0	2	1	4				
1	0	2	1	4	12			
2	0	2	1	4	12	4		
3	0	2	1	4	12	4		7
4	0	2	1	4	12	4	8	7
5	0	2	1	4	11	4	8	7

```
  vertex -> parent
A -> none      E -> G
B -> A         F -> B
c -> A         G -> H
D -> A         H -> F
```

Problem 4

a) We can create a cut such that our two partitions are $\{A, B, D, E, F, Z\}$ and $\{C, Z\}$. With this, we have cut out the edges $\{BC, DC, DZ, GZ\}$. From here, based on the lemma, we call the edge BC light since it has the lowest weight amongst the edges that were cut. We can then say that there exists a MST such that BC is included.

b) By running Kruskal's algorithm, we have too choose the edges that have the least weights, that do not create a cycle. With this, we can see that the first edge chosen is DE, next would be AD and FG. We can see that these edges do not create a cycle. From here, Kurskal's algorithm can continue to find the MST, and we have shown that the edges AD, DE, FG are included within an MST.

Problem 5

a)

```
q = i-1
s = 0
while q % 2 == 0:
   q = math.floor(q/2)
   s += 1

   temp = False
   test = ((2**s) * q)
   if test == i-1: temp = True
```

b) if the number of witnesses is k. If $\frac{1}{4}$ is the probability of a number being prime if there exists a witness. The the probability of the number being composite in that situation is $\frac{3}{4}$. Now for k witnesses, we have it such that the probability for the number n being composite is

$$\left(\frac{3}{4}\right)^k$$