Jakob Roberts - v00484900

Assignment 4

1.

```
n <= length of s
m <=length of t
k <= smallest of m or n
i=0, j=0
while i!=m or j!=n do
    if s[i] == t[j]
        add s[i] to set r
    else if s[i] < t[j]
        i++
    else if s[i] > t[j]
        j++
return set r
```

2.

```
algorithm kmp_search:
    input:
        an array of characters, S (the text to be searched)
        an array of characters, W (the word sought)
    output:
        an integer (the zero-based position in S at which W is found)

    define variables:
        an integer, m ← 0 (the beginning of the current match in S)
        an integer, i ← 0 (the position of the current character in W)
        an array of integers, T (the table, computed elsewhere)

    while m + i < length(S) do
        if W[i] = S[m + i] then
            if i = length(W) - 1 then
                return m
            let i ← i + 1
        else
            if T[i] > -1 then
                let m ← m + i - T[i], i ← T[i]
            else
                let i ← 0, m ← m + 1
    (if we reach here, we have searched all of S unsuccessfully)
    return the length of S
```

3.

I would use the ford-fulkerson algorithm but upon doing each DFS, each path encountered MUST initialize each flow as the minimum. Once all edges are initialized to the minimum, the ford-fulkerson algorithm can be run in it's entirety whilst ensuring that the boundaries of the minimum and maximum flow are not broken.