H.W. 6

Patrick Chen

July 17, 2017

Contents

1

We can run Djikstra's algorithm, except instead of just keeping the total weight required to travel to a node, we keep track of $(level, weight)$. We keep track of both the lowest level required to reach a node and the maximum friendship strength- this can allow up to two stored tuples per node. This way, we make sure that we both keep track of the shortest path as well as ensuring that we reach all nodes k levels away.
The running time off this algorithm is $O(kE + V)$ because with k levels we can only go through the sets of all edges at most once. We only visit each vertex once.

2

a
We do a depth first search. First pick some random library L_0. Then initiate a stack S and a dictionary D with the library names as the keys and 1 representing an installed library and 0 representing an uninstalled library. We add L_0 to the stack. Then we recursively go through the edges of L_0 and add each vertex to the stack until we reach a library with no dependencies. We add that library to the stack, install that library and change the entry in D to 1. Then for all future vertices, we first add the vertex to the stack. Then we check whether the dependencies have been installed. If so, we install the library and change the value in D. Printing the vertices off the top of the stack in order gives an installation order. We visit each vertex and edge at most once, and all other operations are constant, so this takes $O(V + E)$ time. b)

```
def installLibraries(uninstalledList, dependencyDict, installedDict) {
   for library in uninstalledList:
      if installedDict[library] == 'installed':
         pass
      else:
         installLibrary(library, dependencyDict, installedDict)
def installLibrary(libraryName,dependencyDict, installedDict) (
   if libraryName in installedDict:
      return
```

```
  else:
    for dependency in dependencyDict[libraryName]:
      installLibrary(dependency, dependencyDict, installedDict)
```

My algorithm recursively calls installLibrary on dependencies, doing nothing if the dependency has already been installed and calling installLibrary on the dependency if it has not been installed. For any given library, the number of calls to installLibrary is at most D. We also loop through the uninstalledList, which is of length P. All other operations are constant, so our program takes $O(P(1 + D)) = O(P + PD)$ time.