CPSC 490 Problem Set 2

Will Childs-Klein

1

```
def update:
  Input: graph G, graph T, edge (a,b), weight x
  Output: graph T'
  Init: T' = T; G' = G; A,B = {}; curr_min.arg = null
                                  curr_min.val = infiniti
  if (a,b) in T.E, then:
    G'.E = G'.E \ {(a,b)}
    T'.E = T'.E \ {(a,b)}
    A = DFS(G,a)
    B = DFS(G,b)
    for all (j,k) in G'.E s.t. j in A, k in B:
      if w(j,k) < curr_min.val, then:
        curr_min.arg =  (j,k)
        curr_min.val = w(j,k)
    end for
    T'.E = {T'.E} union {curr_min.arg}
  else:
    continue

  return T'
end def

def DFS:
  Input: graph G, vertex s
  Output: set S
  Init: set S = {}; arr visited = [0,0,...,0] of length |G.V|

  visited[s] = 1
  for each edge (s,a) in G.E for given s, a in V:
    if visited[a] equals 0, then:
      recurse DFS(G,a)
    else
      return
```

```
  end for
```

Proof of Correctness

deleting edge breaks G into 2 connected components. we then find the lowest weight edge to reconnect the 2 components. by pre-existing validity of MST, and adding lowest-cost edge to T' that spans A,B, we know that T' is a valid MST.

2

```
def update:
  Input: graph G, graph T, edge (a,b), weight x
  Output: graph C
  Init: graph T' = T; graph C = {}; max.arg = null
                                    max.val = -infiniti
  T'.E = T'.E union {(a,b)}
  C = find_cycle(G)
  for each edge (c,d) in C.E:
    if w(c,d) < max, then:
      max.val = w(c,d)
      max.arg =  (c,d)
  end for
  T'.E = T'.E \ {max.arg}

  return T'
end def

def find_cylce:
  Input: graph G
  Output: returns the graph of an arbitrary cycle in G, which is a subgraph G
end def
```

Proof of Correctness

adding edge introduces exactly one cycle to the graph. we then find the highest weight edge on this cycle and delete it. we're left with valid MST b/c pre-conditions and deleting highest w edge on cycle

3

4