**CS 311 Yoshii HW7 - Graph Algorithms (based on Notes-10B and 11B)**

**DUE: Week 12 Sat**

**TOTAL 34 points Your score:**

**\*NAME: Katie Trinh**

**\*DATE SUBMITTED: 11/21/2023**

**Exercise: Shortest Path Algorithm [16pts]**  **Your score:**

**Use the following directed graph.**

**A: to B is 4 & to F is 2**

**B: to A is 1 & C is 3 & to D is 4**

**C: to A is 6 & to B is 3 & to D is 7**

**D: to A is 6 & to E is 2**

**E: to D is 5**

**F: to D is 2 & to E is 3**

**a) Do Dijkstra's Shortest Path Algorithm starting with C ending with E.** **Trace the algorithm using the same format as I use. Replace ?’s and give the table per step.**

**Initially:**

**Tree has: C**

**Fringe (F\*) has: A6, B3 and D7**

**DistTo of these are initialized to be the edge weights from C.**

**DistTo CandidateEdge**

**T\* C 0 start**

**F\* A 6 from C**

**F\* B 3 from C**

**F\* D 7 from C**

**E inf**

**F inf**

**Step1:**

**Pick B (show that B is T\* in the table)**

**B is next to A1, C3, D4.**

**Note that C is already \*T.**

**Note that A and D are already in Fringe. Should we update them?**

**DistTo to these vertices when going through B:**

* + **DistTO[A] = DistTo[B] + 1 = 4 update (it is better)**
  + **DistTO[D] = DistTo[B] + 4 = 7 no change (it is not better)**

**DistTo CandidateEdge**

**T\* C 0 start**

**F\* A 4 from B updated**

**T\* B 3 from C**

**F\* D 7 from C unchanged**

**E inf**

**F inf**

**Step 2:**

**Pick A**   **(show that it is T\* in the table and highlighted)**

**A is next to: F2, B4 (show that they are F\* if not already)(ignore if already \*T)**

**DistTo to these Fringe (F\*) vertices when going through ??:**

* **DistTO[F] = DistTO[A] + 2 = 6 update (better)**

**(Update the table and indicate updated or unchaged in the table)**

**DistTo CandidateEdge**

**T\* C 0 start**

**F\* A 4 from B updated**

**T\* B 3 from C**

**F\* D 7 from C unchanged**

**E inf**

**F inf**

**Step 3:**

**Pick F**   **(show that it is T\* in the table and highlighted)**

**B is next to: E3, D2 (show that they are F\* if not already)(ignore if already \*T)**

**DistTo to these Fringe (F\*) vertices when going through ??:**

* **DistTO[E] = DistTO[F] + 3 = 9 update (better)**
* **DistTO[D] = DistTO[F] + 2 = 8 no change(not better)**

**(Copy and Update the table and indicate updated or unchaged in the table)**

T\* C 0 start

T\* A 4 from B

T\* B 3 from C

F\* D 7 from C unchanged

F\* E 9 from F updated

T\* F 6 from A

**Step 4:**

**Pick D**   **(show that it is T\* in the table)**

**D is next to: A6, E2 (show that they are F\* if not already) (ignore if already \*T)**

**DistTo to these Fringe (F\*) vertices when going through ??:**

* **DistTO[E] = DistTO[D] + 2 = 9 no change (not better)**

**(Copy and Update the table and indicate updated or unchanged in the table)**

T\* C 0 start

T\* A 4 from B

T\* B 3 from C

T\* D 7 from C unchanged

F\* E 9 from F unchanged

T\* F6 from A

**Step 5:**

**Pick E**   **(show that it is T\* in the table)**

**E is next to: D5 (show that they are F\* if not already) (ignore if already \*T)**

**DistTo to these Fringe (F\*) vertices when going through ??:**

* **DistTO[??] = DistTO[???] + ??? = ???? E is the only vertex to be true and algorithm stops**

**(Copy and Update the table and indicate updated or unchaged in the table)**

T\* C 0 start

T\* A 4 from B

T\* B 3 from C

T\* D 7 from C

T\* E 9 from F

T\* F6 from A

**STOP**

**Stop as soon E becomes \*T.**

**b) Looking at the last table, what was the path from C to E found by the algorithm? Indicate the vertices on the path and the total distance.**

**C->B->A->F->E**

**C = B 3**

**B = A 1**

**A = F 2**

**F = E 9**

**Total distance = 9**

**Program: Implementation DFS of a Graph [2+16=18pts]**  **Your score:**

**Header:**

**Implementation:**

**Client:**

**Test results (required):**

**Total 16 points:**

**Q’s 2 points:**

1. **State of the program statement [2pts]**

* **Did you fix all the errors in HW6 as advised? Yes**
* **Does your program compile without errors? It does not have errors anymore!**
* **List any bugs you are aware of, or state “No bugs”: No bugs!**

**Now that you have a directed graph class from HW6, you can implement DFS.**

**Recall that you did this manually as Week10B In Class EX.**

**You also need a stack class (from HW1) and update it**

**so that you can push vertex names (char) onto a stack.**

**(Where do you need to include stack.h? What do you compile?)**

**You need to add the following 2 functions to dgraph.h and dgraph.cpp:**

* **void visit(int, char) which will enter the given visit number for a given vertex**
  + **this is to indicate the order in which vertices were visited.**
  + **Do not use a loop. Convert A to slot 0, B to slot 1 etc.**
* **bool isMarked(char) which returns true if a given vertex was already visited**

**(0 means not visited)**

* **deDo not use a loop. Convert A to slot 0, B to slot 1 etc.**

**Make sure displayGraph now displays the visit numbers as well.**

**Your client (hw7client.cpp) should implement the DFS algorithm from Notes-10B.doc using the stack class and the graph class functions as follows: (Check hw7.out first)**

**Display the graph before DFS begins.**

**Push A onto the stack to start.**

**While the stack is not empty do:**

**{**

**Remove a vertex v from the stack.**

**Display the vertex name. E.G. “Removed B from stack”**

**If v is not marked yet (visit number is 0) then**

* **mark it (visit it \*\*) and inform the user E.G. “Visit B as 2”**
* **get its adjacency list (slist)**
  + **if no adjacent ones inform the user E.G. “Deadend reached – backup”**
  + **else put adjacent ones on the stack (delete from the rear and push) informing the user**

**else inform the user E.G. “B had been visited already - backup.”**

**Display the stack clearly labeling it as the current stack**

**}**

**Display the Graph nicely with visit numbers for all vertices.**

**(\*\*) visit numbers will start at 1 and increase as you traverse and visit.**

**Requirement:**

* **Make sure you have removed all “trace” messages from llist and slist.**

**(e.g. being in the constructor/destructor)**

* **Make your ouput look nice with indentation etc.**

**Testing:**  **Use the same input file as for HW6.**

**Submit the output for starting at vertex A è Test.txt**

**Note that EC2 includes HW7’s EC for detecting cycles.**

**SUBMIT THESE 5 (or 6) FILES: All files must be commented well!!!**

**If you were told to resubmit HW6 results, submit TestHW6.txt from HW6.**

**Otherwise I will not grade HW7.**

**Must check you have submitted all required files. Cannot submit them later.**

1. **This assignment sheet with your answers.**
2. **dgraph.h**  **-**  **header w/ new functions**
3. **dgraph.cpp**  **- implementation**
4. **hw7client.cpp**  **- client**
5. **Test.txt**  **– script of test results showing what you compiled and ran**

**CHECK LIST before you submit:**

* **Whether working or not, test result must include the lines for compiling your files or we will not grade your program i.e. 0 points for the program.**
* **Did you check your comments and style against CS311 How To Comment.doc??**
* **Did you answer all the questions?**