Program 6 - Flight Paths

**©2019 Larry W. Clark,** this must not be copied to any other website.

This is a continuation of program #5. Combined, pgm#5, pgm#6, and contribution are worth a total of 100 points.

We will use a **double adjacency list graph** to represent locations (airports) and flights:

* Each vertex represents one airport (e.g., SAT, FLL, MCO)
* Edges represent flights from an airport to another
* Since it is a double adjacency list, each flight is two edge nodes. There is an edge node for each of these two lists:
  + For the origin, it is on the successor list.
  + For the destination, it is on the predecessor list
* See cs2123p6.h

**Command File**

Input file stream contains multiple types of records (terminated by EOF). Please use getToken to get the command and then use sscanf to read any associated data. See the assignment#5 for information about other commands.

(Additional commands for Program 6):

MAXSTEPS szOrigin szDest

Prints the maximum number of steps to get from szOrigin to szDest.

PRTALTS szOrigin szDest

For the specified origin and destination, it determines all possible paths (considering time) and prints them.

DELETE szAirport

Deletes the vertex and all flights involving this airport. It should free the edge

nodes that are no longer needed. Flights involving this airport may also be on the

other airport's adjacency lists and need to be deleted.

**Notes**

1. Initially, use the provided data. Additional test data may be provided before the due date.
2. Program #6 non-recursive functions:

void **deleteAirport**(Graph graph, char szAirport[])

This is invoked due to the DELETE command. This will delete an airport:

* Because of the delete, this airport will not be associated with any successors. (This also means that any successors (destination vertex) will no longer have the flight as a predecessor.)
* Because of the delete, this airport will not be associated with any predecessors as the destination vertex. (This also means that no predecessors (origin vertex) will have this flight as a successor.)
* Frees all edge nodes which reference this vertex.
* The airport is marked deleted in the vertex array.
* As part of the extra credit, it is also removed from an overflow chain if it is in the overflow area.

void **prtAlts**(Graph graph, int iOriginVertex, int iDestVertex)

This function prints alternative paths from the origin to the destination:

* Initializes **altpath->iAltCnt** to zero.
* Sets up an empty **path** to pass as an argument to **determinePaths**.
* Invokes **determinePaths** to determine the paths and place them in the global variable **altPath** (of type AltPath).
* Prints the contents of each path from altPath as shown in the example output. See the note about **Global AltPath**.
* When it prints duration, we are simply subtracting the departure time from the arrival time. This value is influenced by the time zone change.
  + Example:
    - Depart SAT at 0600 and arrive in MCO at 1710
    - We will give this a total duration of 11 hrs 10 mins by not considering that 0600 and 1710 were in different time zones. This is to make the calculation easier when considering multiple steps.

1. Program #6 required recursive functions:

int **maxStepsChron**(Graph graph, int iVertex, int iDestVertex, int iPrevArrTm2400)

This is invoked due to the MAXSTEPS command. It determines the maximum number of steps to go from the origin to the destination:

* Initially, iVertex is the origin vertex. On subsequent calls, it is a successor vertex.
* This uses a depth first traversal to return the maximum number of steps to reach the destination.
* Since we have many paths we must set and reset the bVisited flag in this function. If we have already visited a destination in the current path, we won't include the flight in the count.
* This only considers successor flights which occur after the arrival of the previous flight. Please see **Time Considerations** in the Assignment 5 documentation.

void **determinePaths**(Graph graph, int iVertex, int iDestVertex, int iCurStep, Path path

, int iPrevArrTm2400)

This is called from the prtAlts function which you must write. It is similar to maxStepsChron in that it is looking for all valid paths (considering time):

* During traversal to successor flights, it places the current path in the path parameter. (see the include file).
* When it reaches the destination, it copies the current path into the the global variable **altPath**.
* iCurStep is used as a subscript into path->stepM to save an edge pointer.
* We will discuss this in class on 4/23/2019.

1. Program #6 functions for the extra credit (provided by Larry):

int **hash**(Graph graph, char szAirport[])

Hashes a airport to return a number between 0 and g->iOverflowBegin.

void **printHash**(Graph graph)

Prints the hash table in a format that helps show the primary area

and overflow area. For each entry in the primary area, it prints

the entry (if present) and prints corresponding synonyms.

void **printChain**(Graph graph, int iVertex)

1. If a flight already exists on the origin's successorList or the destination's predecessorList, show a warning, but do not terminate.

1. **Hard-coding particular flights** (or **using other data to specify the sequence beyond what I provide**) will result in a **zero** on the entire assignment (Pgm#5 and Pgm #6).

**Global Altpath**

**prtAlt** and **determinePaths** reference a global variable named altPath of type AltPath.

In your file containing your main function:

* To declare the global in the main file (not the main function), insert this statement after your #include statements:

AltPath altPath;

* Insert code for a new function:

AltPath newAltPath()

{

AltPath ap = (AltPath) malloc (sizeof(AltPathImp));

if (ap == NULL)

errExit("could not allcate AltPath");

ap->iAltCnt = 0;

return ap;

}

* Invoke that newAltPath function in your main:

altPath = newAltPath(); // note that the altPath variable was declared above

In your file containing the **prtAlts** and/or **determinePaths** functions, insert this statement after the #include statements:

extern AltPath altPath;

In your **prtAlts** and **determinePaths** functions, you can simply reference **altPath**. Do NOT declare it again.

**How to Compare Times**

Several of the capabilities in program 5 and program 6 need to compare the previous flight's expected completion time with subsequent flight's start time.

* + Previous flight completes at this time in minutes:
    - previousFlight.iTime\*60 + previousFlight.iDurationMins
  + Subsequent flight starts at this time in minutes:
    - subsequentFlight.iTime\*60
  + If the subsequent flight's start time in minutes >= previous flight's completion time in minutes +20 minutes (to give passengers time to catch flights), the subsequent flight can follow the previous flight in a flight path (assuming the previous flight's destination is the subsequent flight's origin). Note that instead of coding 20, use SAFE\_DELTA\_BETWEEN\_FLIGHTS.

**Extra Credit** (8 points + 150/n)

1. Late submissions are not eligible for extra credit.
2. All functionality must be working to receive extra credit.
3. The vertexM array is still used; however, the entries with subscripts before OVERFLOW\_BEGIN are in the primary area. Entries beginning with OVERFLOW\_BEGIN are in the overflow area.
4. A hash function is provided and must be used so that we have consistent results. See the hash.c file that Larry provided.
5. Collisions are chained into an overflow area. A new synonym is placed at the **end of the overflow chain** for an existing overflow chain.
6. Since you are changing how insertions are done, you really don't have a meaningful value for   
   **graph->iNumVertices**. In your code prior to attempting hashing, you have many functions that referenced graph->iNumVertices. You may want to set that variable to **MAX\_VERTICES** to help your existing code work properly.
7. There will be additional data files for the extra credit.
8. Some of the functions that will be changed for extra credit:

* findAirport – it now must hash and understand overflow chains
* your function to insert a flght
* deleteAirport
* newGraph:
  + must initialize every entry in the primary area to have bExists set to FALSE and iHashChainNext to -1 (i.e., our subscript for null).
  + must initialize every entry in the overflow area to create a free list.
  + set iFreehead and iOverflowBegin to OVERFLOW\_BEGIN.

**What to turn in?**

* Team leader (for groups) must turn in the following as a single zip file (named with the group or individual name):
  + all .c files
  + .h file(s)
  + Makefile
  + output
  + instruction to TA sheet (explaining who is on the team (last name, first name) and how to compile/execute your code)
  + no directories in the zip file
* Other participants in a group:
  + instruction sheet explaining who is on the team (last name, first name) and who is submitting the code
  + a document specifying:

|  |  |  |
| --- | --- | --- |
| Source File Name | Primary Responsible Team Member | Reviewer(s) |
|  |  |  |
|  |  |  |

* "selfs" (teams of 1)must turn in the following as a single zip file(named for the individual as lastFirst.zip):
  + all .c files
  + .h file(s)
  + Makefile
  + output
  + instruction to TA sheet (explaining who is on the team (last name, first name) and how to compile/execute your code)
  + no directories in the zip file

Hwk 6.1: via upload in Blackboard (everyone) due when pgm#6 is due

* Group Evaluation form as a PDF
* Peer Evaluation form as a PDF
* What I did form as a PDF

**Program #6 Path Alternatives (Partial Output)**

>> \*

>> \* 6. Considering time, shorter paths

>> \*

>> PRTCHRON SAT

S1-IAH 0600-0630

H3-MCO 1400-1710

H4-LAX 1300-1420

H5-PHX 1500-1600

S2-MCO 0800-1130

S3-ATL 1000-1330

>> PRTALTS SAT MCO

Alternatives for flights from SAT to MCO

Alt TDur Org Dst Fl Dep - Arr Dura

1 11 Hours 10 mins

SAT IAH S1 0600-0630 30

IAH MCO H3 1400-1710 130

2 3 Hours 30 mins

SAT MCO S2 0800-1130 150

>> MAXSTEPS SAT MCO

Maximum chain chron for SAT to MCO contains 2 steps

ooo

>> \*

>> \* 15. Max path from SAT to FLL is 3 steps

>> \*

>> PRTALTS SAT FLL

Alternatives for flights from SAT to FLL

Alt TDur Org Dst Fl Dep - Arr Dura

1 12 Hours 30 mins

SAT IAH S1 0600-0630 30

IAH MCO H3 1400-1710 130

MCO FLL M3 1800-1830 30

2 9 Hours 0 mins

SAT IAH S1 0600-0630 30

IAH FLL H6 1200-1500 120

3 10 Hours 30 mins

SAT MCO S2 0800-1130 150

MCO FLL M3 1800-1830 30

4 8 Hours 30 mins

SAT ATL S3 1000-1330 150

ATL MCO A1 1500-1550 50

MCO FLL M3 1800-1830 30

5 4 Hours 20 mins

SAT FLL S4 1400-1820 200