חורף 2008 מועד א - פתרון

שאלה 1

/\* Topology.h \*/

#ifndef TOPOLOGY\_H

#define TOPOLOGY\_H

typedef struct Topology\_\* PTopology;

typedef void\* PClient;

typedef void\* PAddr;

typedef enum {Success, Failure} Status;

/\* Pointer to function types \*/

typedef PAddr (\*AddrGetter)(PClient);

typedef bool (\*AddrComparator)(PAddr, PAddr);

typedef void (\*ClientDestructor)(PClient);

typedef int (\*PortChooser)(PAddr currentAddr, PAddr destinationAddr);

/\* Interface functions \*/

PTopology TopologyCreate(AddrGetter,AddrComparator,ClientDestructor,PortChooser,int NumPorts); //Returns NULL on failure

void TopologyDestroy(PTopology); //Frees clients' memory

Status TopologyAddClient(PTopology,PClient);

Status TopologyRemoveClient(PTopology,PAddr); //Frees client's memory

PClient TopologyGetClient(PTopology,PAddr); //Returns NULL on failure

Status TopologyAddConnection(PTopology,PAddr Source,int PortNum,PAddr Destination,int Delay);

Status TopologyRemoveConnection(PTopology,PAddr Source,int PortNum);

Status TopologyRoute(int\* Delay,PTopology,PAddr Source,PAddr Destination); //Return value in int\* Delay

#endif

#include "Topology.h"

typedef struct Node\_

{

PClient client;

Node\*\* conArray; //Array of Node\*

int\* conDelay;

Node\* next;

} Node, \*PNode

struct Topology\_

{

//Data fields

int NumPorts;

PNode ListHead;

//Pointers to functions

AddrGetter get\_addr;

AddrComparator compare\_addr;

ClientDestructor destroy\_client;

PortChooser choose\_port;

}

PNode FindNode(PTopology t, PAddr addr)

{

PNode head = t->ListHead;

while (head)

{

if (t->compare\_addr(addr, t->get\_addr(head->client)))

return head;

head = head->next;

}

return NULL;

}

Status TopologyAddConnection(PTopology t,PAddr Source,int PortNum,PAddr Destination,int Delay)

{

if ((PortNum<0)||(PortNum>=t->NumPorts)

return Failure;

PNode src = FindNode(t,Source);

PNode dst = FindNode(t,Destination);

if(!src || !dst)

return Failure;

if (src->conArray[PortNum]!=NULL)

return Failure;

src->conArray[PortNum]=dst;

src->condelay[PortNum]=delay;

return Success;

}

Status TopologyRoute(int\* Delay,PTopology,PAddr Source,PAddr Destination)

{

\*Delay = 0;

PNode curr = FindNode(t,Source);

if (!src)

return Failure;

while (t->get\_addr(curr) != Destination)

{

int port = t->choose\_port(t->get\_addr(curr),Destination);

if ((port<0)||(port>=t->NumPorts)||!curr->conArray[port])

return Failure;

(\*Delay)+=curr->conDelay[port];

curr=conArray[port];

}

return Success;

}

שאלה 2

1. ע"פ סימוני ההנחיה:

tree

לכל צומת ברמה ,  עושה מספר פעולות שהינו לינארי ב"עומק" הרמה (סה"כ אלה השוואות וההחלטה לאיזה בן ללכת). לכן, מספר הפעולות לכל צומת ברמה הינו .

כמו כן, מספר הצמתים ברמה  הינו . לכן הסיבוכיות הכוללת הינה:

,

כאשר האי-שיויון השני נובע כי  לכל , והשיויון בראשון נובע מסכום של טור הנדסי אינסופי.

1. במקרה זה הערימה "נוצרת" ע"י הכנסה סדרתית של איברים ע"י שימוש ב-. כלומר, הסיבוכיות היא:

.

1. כאשר אין פעולת , ניתן להשתמש בפעולות  ו- על מנת "להכניס" איברים לערימה. בעצם כל פעם ניצור ערימה חדשה אשר תכיל את האיבר הנוסף. הסיבוכיות במקרה זה הינה:

.

1. נדרש מערך בגודל , כאשר כל איבר  במערך יצביע לערימה המכילה את כל הבקשות שהתקבלו עד זמן .

פעולת היצירה דומה למתואר בסעיף ג', ולכן סיבוכיותה .

פעולה 2 מתקבלת בסיבוכיות קבוע כי הינה גישה למערך במקום  וקבלת איבר המינימלי בערימה המתאימה.

פעולה 3 מתקבלת ע"י חיפוש פשוט במערך (מעבר על כל האיברים שלו) – סיבוכיות .

פעולה 4: מתקבלת ע"י הוספת איבר חדש לערימה מתאימה – סיבוכיות לוגריתמית.

1. מערך זמן התור הקטן ביותר הינו ממוין בסדר יורד (כי כל פעם מוסיפים איבר נוסף למינימום, ולכן המינימום יכול רק לקטון). המצב הטיפוסי של מערך זה:

array

לכן ניתן פשוט לבצע חיפוש בינארי במערך זה. סיבוכיות החיפוש הבינארי הינה .

פירוט האלגוריתם (חיפוש בינארי סטנדרטי):

נגדיר את המשתנים הבאים:

1. low – אינדקס הגבול הנמוך בתחום החיפוש במערך.
2. high - אינדקס הגבול הגבוה בתחום החיפוש במערך.
3. mid – (low+high)/2

נבצע את הלולאה הבאה:

תנאי: כל עוד low<high נבדוק האם ערכו של האיבר במקום ה- mid שווה ל- mid.

1. אם כן נחזיר את mid כאינדקס המבוקש.
2. אם הוא קטן מהאינדקס, כל החצי העליון של המערך מיותר (לא יכול להיות איבר אשר שווה לאינדקס משום שהמערך ממויין). נקבע אם כן את גבול החיפוש העליון (high) להיות mid-1 ואת mid להיות שוב באמצע בין הגבול התחתון והגבול העליון.
3. אם הוא גדול מהאינדקס מהסבר דומה ניתן לוותר על החצי התחתון של המערך ונגדיר את הגבול התחתון להיות שווה ל- mid+1 ואת mid להיות שוב באמצע בין הגבול התחתון והגבול העליון.

שאלה 3

פתרון (הקטעים ב-bold):

סעיף א':

**class Int : public SimpleItem {**

**int \_data;**

**public:**

**Int(int data) : \_data(data) { }**

**int GetKey() const { return \_data; }**

**};**

סעיף ב':

class Key {

public:

**bool equal(const Key& itm) const {**

**if (!AreSame(\*this, itm))**

**return false;**

**return \*this==itm;**

**}**

**virtual bool operator==(const Key& key) const = 0;**

private:

static bool AreSame(const Key& l, const Key& r) { return (typeid(l) == typeid(r)); }

};

const PItem\* PolyList::GetItem(const Key& key) const

{

PNode\* pIter;

//Check if an PItem with the same key exists:

for (pIter = \_pHead; pIter != NULL; pIter = pIter->\_pNext)

if **(pIter->\_item.GetKey().equal(key))**

return &pIter->\_item;

return NULL;

}

סעיף ג':

(i

**class IntList : public SimpleList**

**{**

**public:**

**bool AddItem(const Int& itm) { return SimpleList::AddItem(itm); }**

**};**

(ii

(a הפונק' החדשות מסתירות את הפונקציות הישנות שמקבלות רפרנס לאיבר כללי SimpleItem.

(b

class Int : public SimpleItem {

int \_data;

Int(int data): \_data(data) { }

public:

int GetKey() const { return \_data; }

**static Int\* create(int data) { return new Int(data); }**

};

שאלה 4

א.

#!/bin/csh

foreach i ($1/\*)

if (-f $i) then

head -1 $i

else if (-d $i) then

$0 $i

endif

end

ב..

#!/bin/csh

#

#courses left for the student to complete

set student = `cat students.dat |grep $1`

set number = ` echo $student |cut -d":" -f4`

set id = ` echo $student |cut -d":" -f1`

set taken = `grep $id \*.course|wc -l `

@ left = $number - $taken

echo $left