Project 2: Cost Constrainted Shortest Paths

CS 401

Due April 29 at the beginning of Class

1 Description

Imagine the following situation. You are trying to travel across the country (from start city s to

destination city d), but have a limited budget. To travel between two cities there may be many

p

ortant differences.

.

ath).

.

ath). We will call this set S [v]

.

ansion and the fact that S [v] contains only non-dominated sig-

natures, we can order the set in strictly increasing order of cost and strictly decreasing order

of traversal time.

.

ath to v discovered in the future will make any of the signatures in S [v] dominated. The

next bullet will clarify this.

.

ath (i.e., the total cost of the

p

and in non-decreasing order of path cost.

. B

An invariant will be that S [v] will never have anything deleted from it. In other words, no

p

s track of the

"frontier". Details follow.

-

ath from the source to v which has cost с and traversal time t.

-

ath).

-

hic).

- U

aths and different choices will have different tradeoffs between cost and time - e.g., taking a bus

may be chea

p

ath signatures are added to the S [] sets in non-decreasing order of

с.

B

aths (signatures) sitting in the

hea

p

reserves

the

p

eration. This is becaue

for a

p

ro

p

The hea

p

The hea

p

ath is dominated

if some other

p

(i.e., it is

lexicogra

p

roblem we will have to store a

set of candidate

p

contains data elements of the form < (с, t), v > which indidates that we have

discovered a

p

Some of these

p

erty that

p

ath is better in one of the dimensions and better (less than) than or equal in

the other (i.e., there is no

p

ecause of this order of ex

p

timal (dom-

inated by some other

p

(a min-hea

p

. When we see a new

p

ath and the total traversal time of the

p

In a manner similar to Dijkstra, we will ex

p

roblem. Part

of your job will be to foramlize it into an actual im

p

oint in using the first

p

ath signatures sitting in the hea

p

ath, we just insert it.

lemtation.

The algorithm has a Dijkstra-like sturture with some im

p

For a vertex v, we only need to store "non-dominated"

p

ensive.

Essentially, what you have then is a doubly weighted gra

p

articular vertex, we may have many candidate

p

In Dijkstra, we store a label d[v] at each vertex. For our

p

aths to v and the "signature" for each such

p

but take a long time; renting a car may be fast, but ex

p

In addition to maintaining S [v] , we also maintain a binary hea

p

nlike Dijkstra's algorithm, we will never do a DecreaseKey ( ) o

p

ath signatures. A

p

which kee

p

h. One weight is the cost с and

the other is the traversal time t. We will assume that all cost and traversal time values are non-

negative integers.

Your goal then is to determine the fastest way to get to your destination within your budget.

2 Algorithm Sketch

This section gives an overview of the main ideas behind an algorithm solving this

p

may turn out to be sub-o

p

y secondarily, we mean that time is used as a tie-breaker in the hea

p

) is ordered first by cost and second by time. This

p

s.

ecall that in the formulation, you have a budget. The above algorithm is the same regardless

of budget, but when it is done, you examine t he non-dominated solutions at the destination and

p

ossible, but this is not required).

encil a n d Pa per

B

aths) and will have four command line

p

ick the fastest one within your budget. (You can use the budget to make the algorithm a bit faster

by sto

pp

-

end it; if it is dominated, you discard it.

-

by a De t e Min (), we need to make sure that it is not dominated by a

p

clearly.

( 2 ) R ecall that cost and time values are non-negative integers.

- L

et the sum of all edge costs be C = AeeG) с( e )

- L

et the sum of all edge times be T = AeeG) t( e )

- L

et M =max( C , T )

- U

retty crude.

- U

erations (insertiona

and deletions).

- U

h,

the source vertex s (an int), the destination vertex d (an int) and a budget (also an int).

You will

p

roblem with at least 5 vertices and at least 3 non-

dominated

p

aths

should overla

p

efore starting your

p

Also unlike Dijkstra's algorithm, when we remove a candidate signature < (с,t),v >

from the hea

p

ing early when

p

roblems.

(1) Construct an exam

p

arameters.

The usage is "c p a th <fi

l

is greater than any entry in S [v] , it must be greater than

the last t-value in S [v] . What does this mean ? You can check if the extracted solution

is dominated by a

p

sing M, give an u

pp

arison with the last entry in S [v] . If it is

not dominated, you a

pp

aths from the s

p

ath cost and time

with those of the edge.

B

aths). Simulate the algorithm by hand. You need

not simulate the individual hea

p

rovide a makefile which

p

.

When is this solution dominated by a solution already in S [v] ? We already know that с

from the newly extracted solution is at least as large as all с's in S [v] . Do any of them

have smaller t-values ? Well, the last entry in S [v] has the smallest t-value in S [v] , so if

the new t-value from the hea

p

se this result to give an u

pp

o

p

sing this bound, give an u

pp

le instance of the

p

(i.e., share common sub

p

er bound on \S[ ] \ . R ecall that all cost values will be distinct (so

will time values). Your bound will be

p

rior solution by one com

p

er bound on the overall runtime of the algorithm.

rogr a m Speci fi cs

Your executable will be called c p a th (for constrained

p

ecified source to the s

p

and"

this solution and add new signatures to the hea

p

, you might as

well do a check to make sure it is not already dominated (as above). The algorithm still

works if you don't do this check.

The above are the key ideas of the algorithm. R ead it multi

p

Only when you have discovered a new non-dominated

p

rogram, you will do the following exercises /

p

roduces the executable c p a th ) .

er bound on the number of hea

p

o

p

. This is done in the natural way by

examining edges leaving the vertex and "augmenting" the current

p

le times if it hel

p

efore inserting this new candidate into the hea

p

erations, just maintain the contents of the hea

p

ath as above, do you "ex

p

ecified destination. Some of the

p

> < s > < d > <bu d g e t> " . So you are given a file contains the gra

p

reviously found solution (i.e., one already in S [v] ). Im portant : recall that S[v] is

ordered in increasing order of с and decreasing order of t and by our invariant, all of the

signatures are non-dominated; now, we have just removed < (с, t), v > from the hea

p

ath exists, the

p

licitly numbered

0 .. V - 1.

After this there is a sequence of edges in the form "( u , v , c , t )" which says there is an edge

from u to v with cost с and traversal time t. For instance "( 5,2,10,5 ) indicates that there is an

edge from vertex to vertex with cost of 10 units and traversal time of units.

Sam

p

rogram. We will use the

tu r nin

p

rogram re

p

osted.

Output

After your

p

le files will be

p

le and clear.

5 Tu rning in y o uprogr a m

In addition to your writeu

p

rogram runs it will re

p

ath

itself (as a sequence of vertices). The format is u

p

rogram on the CS machines. Details will be

p

ort the fastest cost-feasible

p

to you - just kee

p

, you will of course have to turn in your

p

osted to the web.

orts so). You should give the cost and traversal time of the

p

it sim

p

File Format

The first line of the file will contain the number of vertices. V ertices are always im

p

ath (if no feasible

p

ath as well as the

p