Dayou Du
N13825075
dd2645@nyu.edu

Homework 9 - SE
Fundamental Algorithms

2017-11-30

Ex 8.0

a)

Algorithm 1 Ex 8.0a

```
procedure FloydWarshall(n, EdgCst[*,*]; ; PathCst[*,*], Intermediate[*,*])
    for i ∈ [1,n] do
        for j ∈ [1,n] do
            if exist edge (i,j) then
                PathCst[i][j] ← EdgCst[i][j]
                Intermediate[i][j] ← 0
            else
                PathCst[i][j] ← +∞
                Intermediate[i][j] ← −1
    for k ∈ [1,n] do
        for i ∈ [1,n] do
            for j ∈ [1,n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
                    Intermediate[i][j] ← k
```

I got this question CORRECT.

b)

Algorithm 2 Ex 8.0b

```
procedure PathRecoveryDriver(i, j; ; Intermediate[*,*])
    k ← Intermediate[i][j]
    if k == −1 then
        Can NOT Reach
    else
        printPath(i, j, Intermediate)
        print(j)
procedure printPath(i, j; ; Intermediate[*,*])
    k ← Intermediate[i][j]
    if k == 0 then
        print(i)
    else
        printPath(i, k, Intermediate)
        printPath(k, j, Intermediate)
```

I got this question CORRECT.

Ex 8.05

Algorithm 3 Ex 8.05

procedure $FloydWarshall(n, EdgCst[*,*];; Intermediate[*,*], PathCst[*,*])$
  for $i \in [1, n]$ **do**
    for $j \in [1, n]$ **do**
      $CurrCost \leftarrow +\infty$
      $Intermediate[i][j] \leftarrow -1$
      for $k \in [1, n]$ **do**
        if $EdgCst[i][k] + EdgCst[k][j] < CurrCost$ **then**
          $CurrCost \leftarrow EdgCst[i][k] + EdgCst[k][j]$
          $Intermediate[i][j] \leftarrow k$
      $PathCst[i][j] \leftarrow CurrCost$

I got this question CORRECT.
Note: If the original $EdgCst$ DO set the self loop to zero ,then we should set them to $+\infty$ in $TwoEdge$ initialization

Ex 8.1

Algorithm 4 Ex 8.1

procedure $FloydWarshall(n, EdgCst[*,*];; PathCst[*,*], Intermediate[*,*])$
  for $i \in [1, n]$ **do**
    for $j \in [1, n]$ **do**
      if exist edge (i, j) **then**
        $PathCst[i][j] \leftarrow EdgCst[i][j]$
        $Intermediate[i][j] \leftarrow j$
      else
        $PathCst[i][j] \leftarrow +\infty$
        $Intermediate[i][j] \leftarrow -1$
  for $k \in [1, n]$ **do**
    for $i \in [1, n]$ **do**
      for $j \in [1, n]$ **do**
        if $PathCst[i][j] > PathCst[i][k] + PathCst[k][j]$ **then**
          $PathCst[i][j] \leftarrow PathCst[i][k] + PathCst[k][j]$
          $Intermediate[i][j] \leftarrow Intermediate[i][k]$

I got this question CORRECT.

Ex 8.rwb

$$Hedge[i][j] = \begin{cases} Redge[i][j-n] & , i \le n < j \le 2n \\ Wedge[i-n][j-2n] & , n < i \le 2n < j \le 3n \\ Bedge[i-2n][j] & , 2n < i \le 3n, j \le n \\ +\infty & , else \end{cases} \tag{1}$$

After running FW algorithm on $Hedge$, we have $PathCost_G[i][j] = PathCost_H[i][j], i \le n, j \le n$
I got this question CORRECT.

Ex 8.cookie

a)

Algorithm 5 Ex 8.cookie(a)

procedure $FloydWarshall(n, Cookie[*], EdgCst[*,*];; PathCst[*,*], PathCookie[*,*])$
    for $i \in [1, n]$ **do**
        for $j \in [1, n]$ **do**
            if exist edge (i, j) **then**
                $PathCst[i][j] \leftarrow EdgCst[i][j]$
                $PathCookie[i][j] \leftarrow Cookie[i] + Cookie[j]$
            else
                $PathCst[i][j] \leftarrow +\infty$
                $PathCookie[i][j] \leftarrow -1$
    for $k \in [1, n]$ **do**
        for $i \in [1, n]$ **do**
            for $j \in [1, n]$ **do**
                if $PathCst[i][j] > PathCst[i][k] + PathCst[k][j]$ **then**
                    $PathCst[i][j] \leftarrow PathCst[i][k] + PathCst[k][j]$
                    $PathCookie[i][j] \leftarrow PathCookie[i][k] + PathCookie[k][j] - Cookie[k]$

I got this question CORRECT.

b)

Algorithm 6 Ex 8.cookie(b)

procedure $FloydWarshall(n, Cookie[*], EdgCst[*,*];; PathCst[*,*], PathCookie[*,*])$
    for $i \in [1, n]$ **do**
        for $j \in [1, n]$ **do**
            if exist edge (i, j) **then**
                $PathCst[i][j] \leftarrow EdgCst[i][j]$
                $PathCookie[i][j] \leftarrow Cookie[i] + Cookie[j]$
            else
                $PathCst[i][j] \leftarrow +\infty$
                $PathCookie[i][j] \leftarrow -1$
    for $k \in [1, n]$ **do**
        for $i \in [1, n]$ **do**
            for $j \in [1, n]$ **do**
                $kCst \leftarrow PathCst[i][k] + PathCst[k][j]$
                $kCookie \leftarrow PathCookie[i][k] + PathCookie[k][j] - Cookie[k]$
                if $PathCst[i][j] > kCst$ or ($PathCst[i][j] == kCst$ and $PathCookie[i][j] < kCookie$) **then**
                    $PathCst[i][j] \leftarrow kCst$
                    $PathCookie[i][j] \leftarrow kCookie$

I got this question CORRECT.

Ex 8.vb

a)

Let the position of Vienero's bakery is Vertex v, then we can build a new graph H with:

$$Hedge[i][j] = \begin{cases} Ecost[i][j] & , i, j \leq n \text{ , } or\ n < i, j \leq 2n \\ 0 & , i == v\ and\ j == v + n \\ +\infty & , else \end{cases} \tag{2}$$

After running FW algorithm on $Hedge$, we have $PathCost_G[i][j] = PathCost_H[i][j+n], i, j \leq n$
I got this question CORRECT.

b)

$c \approx 8$
Reason: Now we are running a single pass FW algorithm on a graph H, which have $2n$ vertices, thus the running time is $(2n)^3 = 8n^3$
I got this question CORRECT.

Ex 8.cookied

a)

Build a new graph H with $2n$ vertices, and:

$$Hedge[i][j] = \begin{cases} Ecost[i][j] & , i, j \leq n \text{ , } or\ n < i, j \leq 2n \\ 0 & , j == i + n\ and\ Cookie[i] \neq 0 \\ +\infty & , else \end{cases} \tag{3}$$

After running FW algorithm on $Hedge$, we have $PathCost_G[i][j] = PathCost_H[i][j+n], i, j \leq n$
I got this question CORRECT.

b)

Algorithm 7 Ex 8.cookied b

```
procedure FloydWarshallWithCookies(n, EdgCst[∗, ∗]; ; CookiePathCst[∗, ∗])
    create empty helper array PathCst[∗, ∗]
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
            else
                PathCst[i][j] ← +∞
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            CurrCost ← +∞
            for k ∈ [1, n] do
                if PathCst[i][k] + PathCst[k][j] < CurrCost and Cookie[k] ≠ 0 then
                    CurrCost ← PathCst[i][k] + PathCst[k][j]
            CookiePath[i][j] ← CurrCost
```

I got this question CORRECT.

Ex 8.2718

Algorithm 8 Ex 8.2718

```
procedure FloydWarshall(n, EdgCst[∗, ∗]; ; PathCst[∗, ∗], PathEdges[∗, ∗])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
                PathEdges[i][j] ← 1
            else
                PathCst[i][j] ← +∞
                PathEdges[i][j] ← −1
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
                    PathEdges[i][j] ← PathEdges[i][k] + PathEdges[k][j]
```

I got this question CORRECT.

Ex 8.3

The paths that have less or equal than two edges from vertex 1 to vertex 2. The program will find the shortest among them.
I got this question CORRECT.

Ex 8.21

a)

Algorithm 9 Ex 8.21a

```
procedure FloydWarshall(n, EdgCst[∗, ∗]; ; PathCst[∗, ∗], Intermediate[∗, ∗])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
                Intermediate[i][j] ← 0
            else
                PathCst[i][j] ← +∞
                Intermediate[i][j] ← −1
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > max{PathCst[i][k], PathCst[k][j]} then
                    PathCst[i][j] ← max{PathCst[i][k], PathCst[k][j]}
                    Intermediate[i][j] ← k
```

I got this question CORRECT.

b)

Algorithm 10 Ex 8.21b

```
procedure FloydWarshall(n, EdgCst[∗, ∗]; ; PathCst[∗, ∗], Intermediate[∗, ∗])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
                Intermediate[i][j] ← 0
            else
                PathCst[i][j] ← +∞
                Intermediate[i][j] ← −1
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] × PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] × PathCst[k][j]
                    Intermediate[i][j] ← k
```

I got this question CORRECT.

c)

The cycles should not have a path product that less than 1.
The reason is similar to the negative cycle for the original problem: If we have a cycle that the path product is less than 1, then the expand will not stop since we can always get a smaller product through this cycle again and again.
I got this question CORRECT.

Ex 8.2

Algorithm 11 Ex 8.2

```
procedure FloydWarshall(n, EdgCst[*,*];; PathCst[*,*], Intermediate[*,*])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
                Intermediate[i][j] ← i
            else
                PathCst[i][j] ← +∞
                Intermediate[i][j] ← −1
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
                    Intermediate[i][j] ← Intermediate[k][j]
```

I got this question CORRECT.

Ex 8.33

Build a new graph H with $3n$ vertices, and:

$$Hedge[i][j] = \begin{cases} Ecost[i][j] & , i,j \leq n\ , Or\ n < i,j \leq 2n, Or\ 2n < i,j \leq 3n \\ 0 & , i \leq n < j \leq 2n, And\ vertex\ (j-n)\ has\ a\ bank \\ 0 & , n < i \leq 2n < j \leq 3n, And\ vertex\ (j-2n)\ has\ a\ shop \\ +\infty & , else \end{cases} \tag{4}$$

Then, we can run FW algorithm on graph H to calculate $PathCost[s][s+2n]$, or use Dijkstra algorithm to find the shortest path from node s to node $s+2n$
I got this question CORRECT.

Ex 8.39

a)

Build a new graph H with $2n$ vertices, and:

$$Hedge[i][j] = \begin{cases} Ecost[i][j-n] & , i \leq n < j \leq 2n\ \ And\ \ i+n \neq j \\ Ecost[i-n][j] & , j \leq n < i \leq 2n\ \ And\ \ i \neq j+n \\ 0 & , i = j \leq n \\ +\infty & , else \end{cases} \tag{5}$$

Then, we can run FW algorithm on graph H, and $PathCost_G[i][j] = PathCost_H[i][j], i,j \leq n$
The solution is almost CORRECT, but it missed the situation that "0" is also a EVEN NUMBER. Which means, no matter how the original $Ecost$ initialized the "self loop" $Ecost[i][i]$, we should set our $Hedge$ as following:

1. The diagonal elements where index id less or equal than n should be set to zero, because zero is a even number and satisfy the requirement.
2. The diagonal elements where index id greater than n should be set to $+\infty$, otherwise we can follow the self edges back.

b)

Algorithm 12 Ex 8.39b

```
procedure EvenEdge(n, EdgCst[*, *]; ; PathCst[*, *])
    Create helper array TwoEdge[*, *]
    TwoEdge(n, EdgCst, TwoEdge)
    FloydWarshall(n, TwoEdge, PathCst)
procedure TwoEdge(n, EdgCst[*, *]; ; TwoEdge[*, *])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            CurrCost ← +∞
            for k ∈ [1, n] do
                if EdgCst[i][k] + EdgCst[k][j] < CurrCost then
                    CurrCost ← EdgCst[i][k] + EdgCst[k][j]
            TwoEdge[i][j] ← CurrCost
procedure FloydWarshall(n, EdgCst[*, *]; ; PathCst[*, *])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                PathCst[i][j] ← EdgCst[i][j]
            else
                PathCst[i][j] ← +∞
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
```

I got this question CORRECT.
Note: If the original $EdgCst$ DO set the self loop to zero ,then we should set them to $+\infty$ in $TwoEdge$ initialization

c)

Algorithm 13 Ex 8.39c

procedure $FloydWarshall(n, EdgCst[*,*];; OddCst[*,*], EvenCst[*,*])$
  for $i \in [1, n]$ **do**
    for $j \in [1, n]$ **do**
      $EvenCst[i][j] \leftarrow +\infty$
      if exist edge (i, j) **then**
        $Odd[i][j] \leftarrow EdgCst[i][j]$
      else
        $PathCst[i][j] \leftarrow +\infty$
  for $k \in [1, n]$ **do**
    for $i \in [1, n]$ **do**
      for $j \in [1, n]$ **do**
        $EvenMin \leftarrow min\{EvenCst[i][k] + EvenCst[k][j], OddCst[i][k] + OddCst[k][j]\}$
        $OddMin \leftarrow min\{OddCst[i][k] + EvenCst[k][j], EvenCst[i][k] + OddCst[k][j]\}$
        $OddCst[i][j] \leftarrow min\{OddCst[i][j], OddMin\}$
        $EvenCst[i][j] \leftarrow min\{EvenCst[i][j], EvenMin\}$

The algorithm is basically correct. But note that like question a, we have to aware of the "self loops":

1. Zero is also a even number, thus in the final result, $EvenCst[i][i]$ has to be zero.
2. There are possibly self loops with odd number of edges which could change the a route between even or odd.

So the problem is, are we, necessarily, need to compare $OddMin$ with $EvenMin + Odd[k][k]$ or $EvenMin$ with $OddMin + Odd[k][k]$?
The solution wrote the comparation but it didn't give out the necessity of it. Also it didn't prove that why the one without this comparation is incorrect. As I think, this compare is unnecessary. This does not mean that the program will have the exact same temporal result after each cycle, however I think they will have the same and correct final result. Here's a brief proof(Not sure whether it is correct) :
Let's just take look at the comparation between $OddMin$ and $EvenMin + Odd[k][k]$, the other one is symmetry. We first write out the whole equation without $OddMin$ or $EvenMin$:

$$Even[i][k] + Odd[k][j] \tag{6}$$

$$Odd[i][k] + Even[k][j] \tag{7}$$

$$Odd[i][k] + Odd[k][j] + Odd[k][k] \tag{8}$$

$$Even[i][k] + Even[k][j] + Odd[k][k] \tag{9}$$

The full comparation(as is done in the standard solution) will compare all the four equations above. However If calculate equation(6) - equation(8):

$$Even[i][k] - (Odd[i][k] + Odd[k][k]) \tag{10}$$

SO IF the $Even$ and Odd array still comply the similar definition of the $PathCst$ array in the original problem, then equation(6) is always less or equal than equation(8). And also equation(7) is less or equal than equation(9).

Ex 8.41

Algorithm 14 Ex 8.41

```
procedure FloydWarshall(n, EdgCst[*,*]; ; First[*,*], Second[*,*])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            Second[i][j] ← +∞
            if exist edge (i, j) then
                First[i][j] ← EdgCst[i][j]
            else
                First[i][j] ← +∞
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if First[i][j] > First[i][k] + First[k][j] then
                    Second[i][j] ← min{First[i][j], First[i][k] + Second[k][j], Second[i][k] + First[k][j]}
                    First[i][j] ← PathCst[i][k] + PathCst[k][j]
                else
                    Second[i][j] ← min{Second[i][j], First[i][k] + First[k][j]}
```

I got this question CORRECT.

Ex 8.42

Algorithm 15 Ex 8.42

```
procedure FloydWarshall(n, EdgCst[*,*]; ; PathCst[*,*], PathNum[*,*])
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            if exist edge (i, j) then
                First[i][j] ← EdgCst[i][j]
                PathNum[i][j] ← 1
            else
                First[i][j] ← +∞
                PathNum[i][j] ← 0
    for k ∈ [1, n] do
        for i ∈ [1, n] do
            for j ∈ [1, n] do
                if PathCst[i][j] > PathCst[i][k] + PathCst[k][j] then
                    PathCst[i][j] ← PathCst[i][k] + PathCst[k][j]
                    PathNum[i][j] ← PathNum[i][k] × PathNum[k][j]
                else
                    if PathCst[i][j] == PathCst[i][k] + PathCst[k][j] then
                        PathNum[i][j] ← PathNum[i][j] + PathNum[i][k] × PathNum[k][j]
```

I got this question CORRECT.