Solution to Assignment 2

Thomas Feng *

February 20, 2003

1. *Solution:*

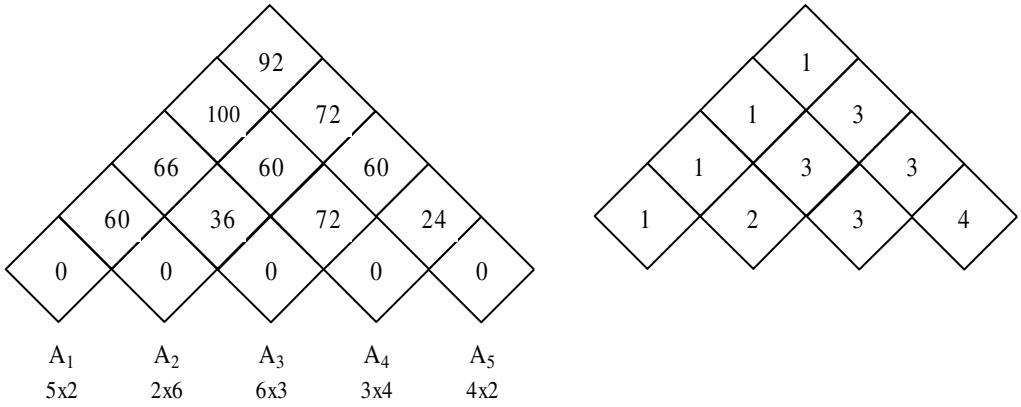


Figure 1: m and s tables

2. *Solution:*

i. Use brute-force way to enumerate all possible subsets

```
function fs ( S )
  return fs_rec ( sum ( S ) / 2 , S )

function sum ( S )
  t = 0
  for every s in S
    t = t + w(s)
  return t

function fs_rec ( sum , S )
  if ( sum ( S ) < sum )
    return false
  elif ( sum ( S ) = sum )
    return S
  for every s in S
    res = fs_rec ( sum , S - {s} )
```

*Email: thomas@email.com.cn
Homepage: http://moncs.cs.mcgill.ca/people/tfeng/

```
    if ( res != false)
      return res
  return false
```

ii. Use two dimentional table $R^{n\times w}$ (s is the maximal total weight)

```
function sum ( S )
  t = 0
  for every s in S
    t = t + w(s)
  return t

function fs ( S )
  w = sum ( S )
  if ( w is odd )
    return false
  else
    w = w / 2
  initialize table T[0..n, 0..w]
  initialize table G[0..n, 0..w]
  set T[0, k] to 0 , k = 0, 1, ..., w
  for i = 1 to n
    for j = 0 to w
      if ( j < S[i] )
        T[i, j] = T[i-1, j]
        G[i, j] = j
      else
        a = T[i-1, j]
        b = T[i-1, j-S[i]] + S[i]
        if a > b
          T[i, j] = a
          G[i, j] = j
        else
          T[i, j] = b
          G[i, j] = j-S[i]
  return get_result ( T , G ,  w )

function get_result ( T , G , w )
  result = {}
  for i = n downto 0
    if ( T [i, w] = w )
      add S[i] to result
      w = w - S[i]
  if ( result is empty )
    return false
  else
    return result
```

iii. Discussion: solution (i) usually takes $O(n^n)$ time, which, though large, is still acceptable for this NP-hard problem; solution (ii) does not work if w is very large or fractional.

3. (a) *Solution:*
Figure 2 shows such an example:

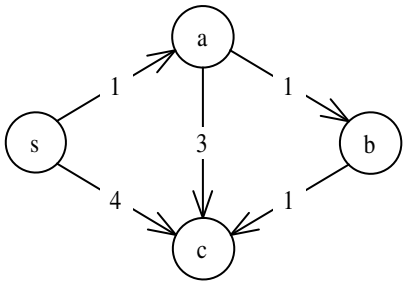


Figure 2: Example to show $T(i,v) \neq d_i(v)$

$T(i,v)$, $i=0$	$T(0,s)=0$	$T(1,a)=\infty$	$T(0,b)=\infty$	$T(0,c)=\infty$
$T(i,v)$, $i=1$	$T(1,s)=0$	$T(1,a)=1$	$T(1,b)=\infty$	$T(1,c)=4$
$d(v)$, *iteration 0*	$d_0(s)=0$	$d_0(a)=\infty$	$d_0(b)=\infty$	$d_0(c)=\infty$
$d(v)$, *iteration 1*	$d_1(s)=0$	$d_1(a)=1$	$d_1(b)=2$	$d_1(c)=3$

The order of the edges considered is: $s \to a$, $a \to b$, $b \to c$, $a \to c$, $s \to c$.

So at the end of the first iteration, $T(1,v) \neq d_1(v)$, where $n=5$.

3. (b) *Proof:*

- For $i=0$, $T(i,s)=d(s)=0$, $T(i,v)=d(v)=\infty, \forall v \neq s$.
- Suppose at the end of iteration i, $T(i,v) \geq d_i(v)$.
  Then, at then end of iteration $i+1$,

$$
\begin{aligned}
T(i+1,v) &= \min\{T(i,v), \min_{(u\neq v)}\{T(i,u)+w(u,v)\}\} && (1)\\
&\geq \min\{d_i(v), \min_{(u\neq v)}\{d_i(u)+w(u,v)\}\} && (2)\\
&\geq \min\{d_i(v), \min_{(u<v)}\{d_{i+1}(u)+w(u,v)\}, \min_{(u>v)}\{d_i(u)+w(u,v)\}\} && (3)\\
&= d_{i+1}(v) && (4)
\end{aligned}
$$

  In (3), the inequation $u < v$ means the value of $d_i(u)$ is got before $d_i(v)$ in each iteration; and vice versa. I use the face $d_i(v) \leq d_{i+1}(v)$.

■

3. (c) *Solution:*
The graph is shown in Figure 3.

In this case, $n=4$. Suppose each iteration of both algorithms considers the edges in such an order: $b \to s$, $s \to c$, $s \to a$, $a \to b$.

$T(i,v)$, $i=0$	$T(0,s)=0$	$T(1,a)=\infty$	$T(0,b)=\infty$	$T(0,c)=\infty$
$T(i,v)$, $i=1$	$T(1,s)=0$	$T(1,a)=1$	$T(1,b)=\infty$	$T(1,c)=1$
$T(i,v)$, $i=2$	$T(2,s)=0$	$T(2,a)=1$	$T(2,b)=2$	$T(2,c)=1$
$T(i,v)$, $i=3$	$T(3,s)=-1$	$T(3,a)=1$	$T(3,b)=2$	$T(3,c)=1$
$d(v)$, *iteration 0*	$d_0(s)=0$	$d_0(a)=\infty$	$d_0(b)=\infty$	$d_0(c)=\infty$
$d(v)$, *iteration 1*	$d_1(s)=0$	$d_1(a)=1$	$d_1(b)=2$	$d_1(c)=1$
$d(v)$, *iteration 2*	$d_2(s)=-1$	$d_2(a)=0$	$d_2(b)=1$	$d_2(c)=0$
$d(v)$, *iteration 3*	$d_3(s)=-2$	$d_3(a)=-1$	$d_3(b)=0$	$d_3(c)=-1$

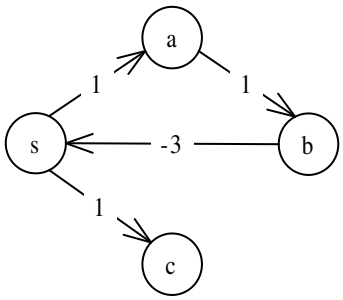


Figure 3: Example to show $T(n-1,v) \neq d_{n-1}(v)$

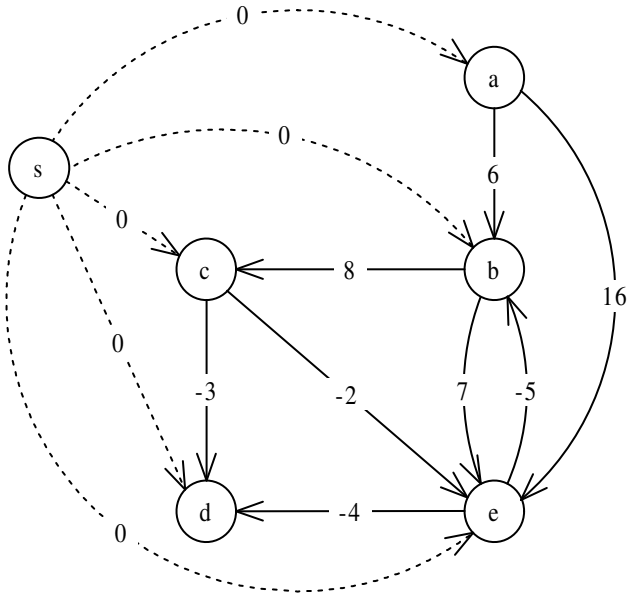


Figure 4: Example to show $T(n-1,v) \neq d_{n-1}(v)$

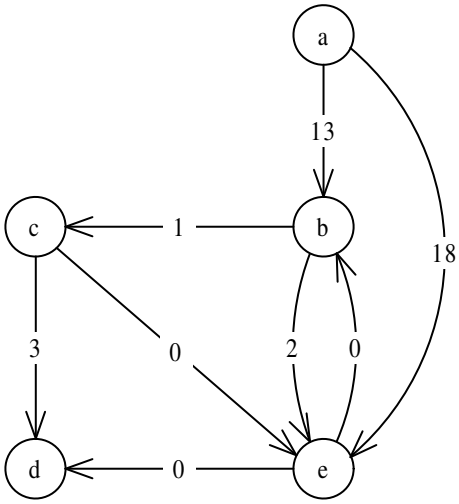


Figure 5: Reweighed graph

4. (a) *Solution:*
In Figure 4, vertex s is added. Below is the table of the execution of Bellman-Fort's algorithm. (Use $T(i,v)$ here.)

$T(i,v)$, $i=1$	$T(1,s)=0$	$T(1,a)=0$	$T(1,b)=0$	$T(1,c)=0$	$T(1,d)=0$	$T(1,e)=0$
$T(i,v)$, $i=2$	$T(2,s)=0$	$T(2,a)=0$	$T(2,b)=-5$	$T(2,c)=0$	$T(2,d)=-4$	$T(2,e)=-2$
$T(i,v)$, $i=3$	$T(3,s)=0$	$T(3,a)=0$	$T(3,b)=-7$	$T(3,c)=0$	$T(3,d)=-6$	$T(3,e)=-2$
$T(i,v)$, $i=4$	$T(4,s)=0$	$T(4,a)=0$	$T(4,b)=-7$	$T(4,c)=0$	$T(4,d)=-6$	$T(4,e)=-2$
$T(i,v)$, $i=5$	$T(5,s)=0$	$T(5,a)=0$	$T(5,b)=-7$	$T(5,c)=0$	$T(5,d)=-6$	$T(5,e)=-2$

4. (b) *Solution:*
From 4. (a) we know $h(a)=0$, $h(b)=-7$, $h(c)=0$, $h(d)=-6$, $h(e)=-2$. So we get the new weight function $\hat{w}(u,v)$:

$$\hat{w}(a,b)=13,\ \hat{w}(a,e)=18,\ \hat{w}(b,c)=1,\ \hat{w}(b,e)=2,\ \hat{w}(e,b)=0,\ \hat{w}(e,d)=0,\ \hat{w}(c,d)=3,\ \hat{w}(c,e)=0$$

4. (c) *Solution:*
Solution is given in Figure 6 (next page).

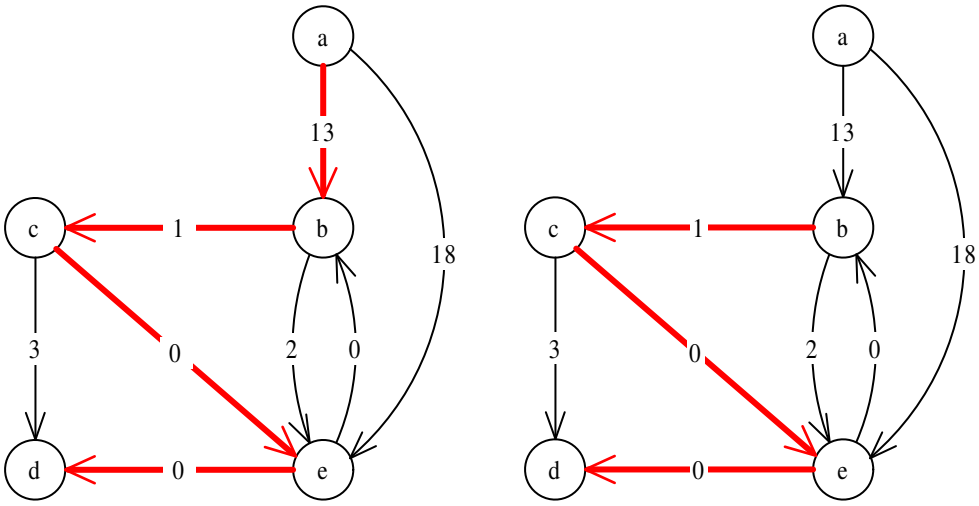
4. (d) *Solution:*
All-pairs shortest path for the graph is given below:

$\hat{\delta}(u,v)$	a	b	c	d	e
a	0	13	14	14	14
b	∞	0	1	1	1
c	∞	0	0	0	0
d	∞	∞	∞	0	∞
e	∞	0	1	0	0

$\delta(u,v)$	a	b	c	d	e
a	0	6	14	8	12
b	∞	0	8	2	6
c	∞	-7	0	-6	-2
d	∞	∞	∞	0	∞
e	∞	-5	3	-4	0

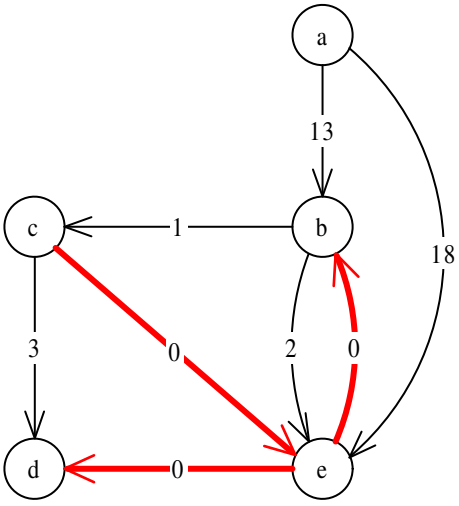
Note: Most of the students use Bellman-Ford algorithm on vertex a in the beginning, without giving any reason why. This is actually wrong! The reason is, not any vertex can do. Suppose instead of a, one takes b, c, d or e, after the reweighing, the edges $a \to b$ and $a \to e$ are deleted, simply because the Bellman-Ford algorithm returns ∞ for the shortest path to a.

For those who noticed this by the knowledge from the textbook, I give a bonus of 5 marks.

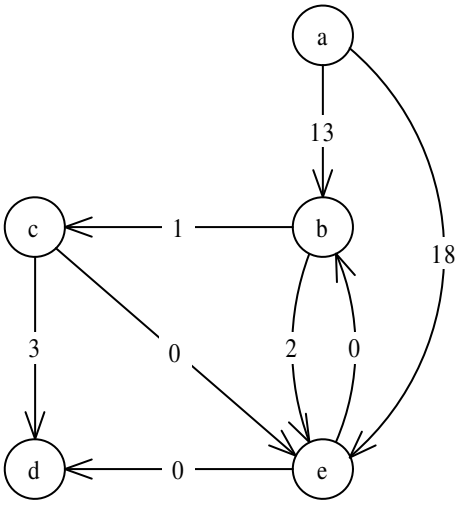


Shortest paths starting from a

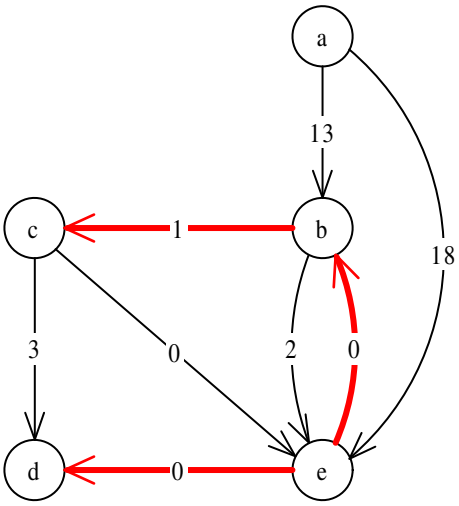
Shortest paths starting from b



Shortest paths starting from c



Shortest paths starting from d



Shortest paths starting from e

Figure 6: Dijkstra trees for all vertices