Problem Set 13.

181240045 潘兔之

Problem 1:

(a) Prove by contradiction.

① Suppose ϕ is complete for P with respect to poly-time reductions. let $L=\phi$. $L'=\{0,1\}^*$, from the problem, we know $L\in P$ and $L'\in P$.

$\Rightarrow L'\le_p L \Rightarrow$ there exists a poly-time computable function f, s.t. for all $x\in\{0,1\}^*$, $x\in L'$ iff $f(x)\in L$

$\Rightarrow$ Notice $L'=\{0,1\}^*$, $x\in L'$ is always true $\Rightarrow f(x)\in L$ is always true

$\Rightarrow$ contradiction, since $L=\phi$

② Suppose $\{0,1\}^*$ is complete for P with respect to poly-time reductions. Let $L=\{0,1\}^*$, $L'=\phi$

$\Rightarrow L'\le_p L \Rightarrow$ existing a poly-time function f'. s.t for all $x\in\{0,1\}^*$, $x\in L'$ iff $f(x)\in L$

$\Rightarrow$ Notice $x\in L'$ is always false $\Rightarrow f'(x)\in L$ is false

$\Rightarrow$ contradiction, since $L=\{0,1\}^*$, $f'(x)\in L$ must be true.

③ Consider a language $L\in P$ which is neither ϕ nor $\{0,1\}^*$, and $x_1\in L$, $x_2\notin L$.

For all $L'\in P$, we have a reduction function

$$f(x)=\begin{cases} x_1, & x\in L' \\ x_2, & x\notin L' \end{cases} \quad \text{for all } x\in\{0,1\}^*$$

Notice $L'\in P$, $f(x)$ can be computed in polynomial time. And $x\in L'$ iff $f(x)\in L$

$\Leftrightarrow L'\le_p L \Rightarrow L$ is complete for P with respect to poly-time reductions

①②③ $\Rightarrow$ proved

(b) L is complete for NP $\Leftrightarrow L\in NP$, $L'\le_p L$ for all $L'\in NP \Rightarrow$ there exists a poly-time reduction function f, s.t. for all $x\in\{0,1\}^*$, $x\in L'$ iff $f(x)\in L$

$\Leftrightarrow x\notin L'$ iff $f(x)\notin L$

$\Leftrightarrow x\in \bar{L}'$ iff $f(x)\in \bar{L}$

since $L, L'\in NP \Rightarrow \bar{L}, \bar{L}'\in coNP$, so we have $\bar{L}'\le_p \bar{L}$ for all $\bar{L}'\in coNP$

$\Rightarrow \bar{L}$ is complete for coNP

Problem 2.

(a) Suppose the boolean formula is $A[0 \cdots 2n]$, where $A[i] \in \{T, F\}$ when i is even, and $A[i]$ is boolean operators when i is odd. Let the known alg. which can decide boolean formula satisfiability be DECIDE(s), where s is a boolean formula. The alg will return True if s is satisfiable. Otherwise it will return False.

10

~~alg. i=0~~

~~while i ≤ 2n-2:~~

~~assign boolean value to A[i] according to A[i+1] and DECIDE(A[i+2, ..., 2n-1])~~

~~i = i+2~~

~~if DECIDE(~~

alg. target = True

i = 0

while $i \leq 2n-2$:

if DECIDE(A[i+2, ..., 2n-1]) == True:

assign boolean value to A[i] s.t. A[i]A[i+1]T = target

target = True

else:

assign boolean value to A[i] s.t. A[i]A[i+1]F = target

target = False

i = i+2

A[2n] = target

Suppose the time-complexity of DECIDE is $O(n^k)$, the total time is $n * O(n^k) = O(n^{k+1})$, which is polynomial.

(b) For a clause in a boolean formula (XVY), there must be at least one variable be True in X and Y, i.e. if X(Y) is False, Y(X) must be True.

Construct a directed graph as following: suppose there are totally n boolean variables $x_1, x_2, \cdots, x_n$, allocate 2n vertices corresponding to $x_1, \neg x_1, x_2, \neg x_2, \cdots, x_n, \neg x_n$. For each clause in the formula, add edges $(\neg m, n)$ and $(\neg n, m)$ to the directed graph.

$(m \vee n)$

if m = True, then n can be either False or True. If m = False, n must be True.

~~So for each edge (u,v) in the graph, there are only three conditions~~

So for each edge $\langle u,v\rangle$ in the graph, there are only three conditions with respect to the boolean value: (True, True), (False, False), (False, True).

We have the following deduction.

The formula is unsatisfiable iff

1° there exists a path from x_i to $\neg x_i$

2° there exists a path from $\neg x_i$ to x_i

Proof. Considering an assignment of $x_1, x_2, \cdots, x_n$, and there is a path from x_i to $\neg x_i$ and $\neg x_i$ to x_i

1° if X_i = True, from the above conclusion, the vertices along the path must be True as well

$\Rightarrow \neg X_i$ = True

$\Rightarrow$ contradiction

2° if X_i = False, considering the path from $\neg X_i$ to X_i, we can ~~prove~~ find contradiction just as the former part

$\Rightarrow$ valid assignment does not exist if there are pathes from x_i to $\neg x_i$ and $\neg x_i$ to x_i

alg. Build a directed graph according to the formula

```
for i=1 to n:
    use BFS to find if there is a path from xi to ¬xi
    use BFS to find if there is a path from ¬xi to xi
    if find path from xi → ¬xi and ¬xi → xi:
        return False
return ~~False~~ True
```

time complexity is $2n \cdot O(n^2) = O(n^3)$, which is polynomial time

Problem 3.

① 0-1 integer programming $\in NP$.

- $\forall$ integer n-vector x with elements in the set $\{0,1\}$ can be a certificate
- matrix A multiplies x takes $O(mn)$ time, so ~~we can~~ 'yes' instances can be verified in polynomial time.

② 0-1 integer programming $\in NP$-hard.

Try to show $3\text{-SAT} \leq_p$ 0-1 integer programming.

Considering a CNF formular ϕ of n variables and m clauses, we construct A and b in the following:

设所有的 boolean variables 为 $a_1, a_2, \cdots, a_n$。

$$\Rightarrow A[i][j] = \begin{cases} 0, & \text{the } i^{th} \text{ clause doesn't contain } a_j \text{ or } \neg a_j \\ -1, & a_j \text{ is in the } i^{th} \text{ clause} \\ 1, & \neg a_j \text{ is in the } i^{th} \text{ clause} \end{cases}, \quad 1 \leq i \leq m, \ 1 \leq j \leq n$$

10

$$b[i] = -1 + \sum_{j=1}^{n} \max(0, A[i][j])$$

Now prove there exists an integer n-vector x s.t. $Ax \leq b \Leftrightarrow \phi$ is satisfiable.

Considering vector x:

$$x[j] = \begin{cases} 1, & a_j \text{ is assigned True} \\ 0, & a_j \text{ is assigned False} \end{cases}, \quad 1 \leq j \leq n$$

"$\Rightarrow$": Considering the i^{th} clause, $y[i] = \sum_{j=1}^{n} A[i][j] \cdot x[j]$

设 $y = Ax$

= num of '$\neg$ True' in i^{th} clause + (− num of 'True' in i^{th} clause)

$b[i] = -1 +$ num of '$\neg$' in i^{th} clause

$Ax \leq b \Rightarrow$ in the i^{th} clause: num of '$\neg$ True' − num of 'True' $\leq -1 +$ num of '$\neg$'

$\Rightarrow$ − num of 'True' − (num of '$\neg$' − num of '$\neg$ True') ≤ -1

$\Rightarrow$ − num of 'True' − num of '$\neg$ False' ≤ -1

$\Rightarrow$ num of 'True' + num of '$\neg$ False' ≥ 1

$\Rightarrow$ i^{th} clause is True $(1 \leq i \leq m) \Rightarrow \phi$ is satisfiable

2° "⇐": ϕ is satisfiable $\Rightarrow \forall$ clause $\in \phi$ is True $\Rightarrow$ num of literals which is True

Notice num of literals which is True = num of 'True' + num of '$\neg$ False' ≥ 1

Considering the i^{th} clause,

num of 'True' + num of '$\neg$ False'

= num of 'True' + (num of '$\neg$' − num of '$\neg$ True')

= num of 'True' − num of '$\neg$ True' + num of '$\neg$' ≥ 1

$\Rightarrow$ − num of 'True' + num of '$\neg$ True' ≤ -1 + num of '$\neg$'

$\Rightarrow \sum_{j=1}^{n} A[i][j] \cdot x[j] \leq b[i] \quad (1 \leq i \leq m)$

$\Rightarrow Ax \leq b$

Problem 4:

(a) Split each $v \in V$ into two vertices, and the edge capacity between them is $l(v)$.

alg.

~~for each $v \in V$:~~

~~for each $(u,v) \in E$:~~

~~$C(u,x) = C(u,v)$~~

~~for each $(v,w) \in E$:~~

~~$C(y,w) = C(v,w)$~~

~~$C(x,y) = l(v)$~~

$V' = \phi$, $E' = \phi$

for each $v_i \in V$:

for each $(v_j, v_i) \in E$:

if $(y_j, x_i) \notin E'$:

$C(y_j, x_i) = C(v_j, v_i)$

$E' = E' \cup \{(y_j, x_i)\}$

for each $(v_i, v_k) \in E$:

if $(y_i, x_k) \notin E'$:

$C(y_i, x_k) = C(v_i, v_k)$

$E' = E' \cup \{(y_i, x_k)\}$

$C(x_i, y_i) = l(v_i)$

$E' = E' \cup \{(x_i, y_i)\}$

$V' = V' \cup \{x_i, y_i\}$

$\Rightarrow$ vertices: $2|V|$

edges: $|E| + |V|$

(b) From the definition of flow network, each vertex lies on some path from the source to the sink

$\Rightarrow$ G contains a path $s \rightsquigarrow v \rightsquigarrow t$

Notice ~~edge~~ f contains edge (v, s) entering s

$\Rightarrow$ f must contain a cycle, and s, v are both in it, i.e. $s \rightsquigarrow u \rightsquigarrow t$ (with $u \to v \to s$)

$\Rightarrow$ reduce the flow value of each edge in that cycle (by 1) the new flow is f'.

Proof: 1° $f(v,s) = 1 \Rightarrow f(v,s) - 1 = 0 \Rightarrow f'(v,s) = 0$

$$2° \quad |f'| = \sum_{u \in V} f'(s,u) - f'(v,s) = \left(\sum_{u \in V} f(s,u) - 1\right) - 0 = \sum_{u \in V} f(s,u) - f(v,s) = |f|$$

alg. Use DFS to find a cycle which is in f and contains s, v

reduce the flow value of each edge in the cycle by 1

the new flow is f'

time complexity. $O(|E|)$

(c) $\forall$ path in G' from $s \to t$ is '$s \to$ one vertex in $L \to$ one vertex in $R \to t$', and there are edges from L to R in G'.

Considering the residual network G'_f, there are edges from L to R and edges from R to L in G'_f. To find the upper bound on the length of any augmenting path, we should make an argumenting path travel between L and R as many as possible.

Suppose there are n vertices in L: $l_1, l_2, \dots, l_n$

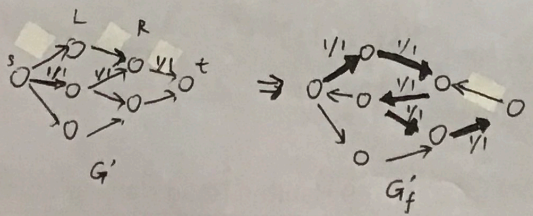
m vertices in R: $r_1, r_2, \dots, r_m$ $(m \le n)$

path is: $s \to l_1 \to r_1 \to l_2 \to r_2 \to \dots \to l_m \to r_m \to t$.

$\Rightarrow$ length is $2m + 1$.

Generally, length is $2 \cdot \min\{|L|, |R|\} + 1$

example:



Problem 5:

alg. Construct a connected graph $G'=(V', E')$, where

$V'=V$, $E'=\{(u,v),(v,u) \mid \text{for each } (u,v)\in E\}$, $c(u,v)=1$ for each $(u,v)\in E'$

connectivity $=\infty$

for each $v_i \in V'$:

set v_i to be the target and an arbitary vertex $u \neq v_i$ be the source.

f_i = EdmondskarpMaxFlow(G', u, v_i)

connectivity = min{connectivity, f_i}

return connectivity

Correctness: Find the minimum number of edges that must be removed to disconnect the graph

$\Leftrightarrow$ Find the minimum number of edges that must be removed to isolate a vertex

$\Leftrightarrow$ Find the min-cut of the flow network where the source and target are determined as the alg.

$\Leftarrow$ Find the max-flow

Problem 6:

$G=(V,E)$:

1° 首先构造一个 bipartite graph $V=L\cup R$, where $L=\{s_1, s_2, \cdots, s_n\}$,

$R=\{d_1, d_2, \cdots, d_k\}$. Mention that $s_i\in L$ represents doctor i's vacation days S_i,

$d_j\in R$ represents the j^{th} vacation period D_j.

$E=\{(s_i, d_j) \mid s_i\in L, d_j\in R, S_i$ include certein days from vacation period $D_j\}$

2° Now consider corresponding flow network G' of G with source vertex s and target vertex t, i.e. $G'=(V', E')$, where $V'=\{s,t\}\cup V$, $E'=\{(s,s_i) \mid 1\le i\le n\}\cup\{(d_j,t) \mid 1\le j\le k\}\cup$

And the capacity of (s,s_i) is $\min\{|S_i|, c\}$, $1\le i\le n$;

the capacity of $(s_i,d_j)\in E$ is 1;

the capacity of (d_j,t) is $|D_j|$, $1\le j\le k$.

3° We just need to find the max flow of G', which we write as f. if $|f|=\sum_j |D_j|$

return True. Otherwise, return False.

Problem 7:

(a) if ~~one~~ the capacity of one edge is increased by 1

⇒ value of the max flow may remain unchanged or strictly increased by 1

⇒ we just need to modify based on the current maximum flow, since no other flow can be greater than current flow even if ~~one~~ capacity of one edge is increased by 1

⇒

alg. Construct residual network G_f of G after capacity of e is increased by 1

while (there is an augmenting path p in G_f),

Augment flow f along the shortest path p

return f

Time complexity,

Notice the value of flow can only increase by 1 at most

⇒ the while-loop will be executed once at most

⇒ Finding an augmenting path cost $O(|V|+|E|)$ time

⇒ Time complexity is $O(|V|+|E|)$

(b) Suppose the capacity of e is c(e), then c'(e) = c(e) - 1 after the decrement

if f(e) = c(e). // f(e) = 0 if edge e is not in the flow

use BFS to find a path p in G from s→t containing e

for edge e' in p:

if f(e') ≠ 0:

f(e') = f(e') - 1

construct residual network G_f of G

while (there is an augmenting path in G_f):

Augment flow f along the shortest path

return f