上 海 交 通 大 学 试 卷（A卷）

（ 2013 至 2014 学年 第 2 学期 ）

学　　号：　　　　　　　　姓名：

课程名称：　算法设计与分析　　　成绩：

1. (10 points) Write the dual to the following linear program.

$$\text{Max } 6x - 4z + 7$$
$$3x - y \leq 1$$
$$4y - z \leq 2$$
$$x, y, z \geq 0$$

Is the solution $(x, y, z) = (1/2, 1/2, 0)$ optimal? Write the dual program of the given linear program and find out its optimal solution.

Sol: Dual program:

Min x+2y+7

subject to $3x \geq 6$

$-x+4y \geq 0$

$-y \geq -4$

$x, y \geq 0$

(x, y, z) = (1/2,1/2,0) is optimal.

我承诺，我将严格遵守考试纪律。

承诺人：__________

题号									
得分									
批阅人(流水阅卷教师签名处)									

2. (10 points) A **Minimum Makespan Scheduling** problem is as follows:

Input processing times for n jobs, $p_1, p_2, \ldots, p_n$, and an integer m.

Output an assignment of the jobs to m identical machines so that the completion time is minimized. We know that by a greedy approach on the problem, the approximation factor 2. Give a tight example to show the approximation guarantee.

Sol:

Algorithm:

1. Order the jobs arbitrarily.

2. Schedule jobs on machines in this order, scheduling the next job on the machine that has been assigned the least amount of work so far.

A tight example for this algorithm is provided by a sequence of m^2 jobs with unit processing time, followed by a single job of length m. The schedule obtained by the algorithm has a makespan of $2m$, while $OPT = m + 1$.

3. (10 points) **Steiner Forest Problem** is defined as follows,

Input an undirected graph $G=(V,E)$, nonnegative costs $c_e \geq 0$ for all edges $e \in E$ and k pairs of vertices $(s_i,t_i) \in V$.

Output a minimum cost subset of edges F such that every (s_i,t_i) pair is connected in the set of selected edges.

Represent the problem as an integer program.

Sol: Define $r(u,v)=\begin{cases}1 & \text{if } \exists \text{i s.t. } (u,v)=(s_i,t_i) \\ 0 & \text{o.w.}\end{cases}$ and let $(S,\overline{S})$ is a cut of G; then define:

$$f(S)=\begin{cases}1 & \text{if } \exists u \in S \text{ and } v \in \overline{S} \text{ s.t. } r(u,v)=1 \\ 0 & \text{o.w.}\end{cases}$$

Then the Steiner Forest Problem can be rewritten:

$$\min \sum_{e\in E} c_e x_e$$

$$\textit{subject to} \quad \sum_{e:e\in\delta(S)} x_e \geq f(S) \quad S \subseteq V$$

$$x_e \geq 0 \qquad e \in E$$

4. (15 points) Given a reduction from the **Clique Problem** to the following problem, which you also need to show to be a search problem.

Input a undirected graph *G* and a positive integer k.

Output a Clique of size k as well as an Independent Set of size k, provided both exist.

Sol: 1. Since given an answer of the problem, we can check whether the given clique and independent set are correct in polynomial time, so it's a search problem.

2. We now give a reduction from the Clique Problem to this problem:

Clique problem: Given a undirected graph G and a positive integer k, ask whether the graph G has a clique of size k.

The reduction is very simple, we just construct the graph $G = (V \cup V', E)$, where $|V'| = k$. Then if G' has a clique of size k > 1, all the nodes are in the set V. So G has a clique of size k if and only if G' has both a clique of size k and an independent set of size k. Since the reduction is in polynomial time, we have finished the reduction.

5. (15 points) Given an undirected graph $G = (V, E)$ in which each node has degree $\leq d$, find an approximation algorithm for maximal **independent set** with the factor $1/(d+1)$.

Sol:

```
GREEDY(G):
S ← ∅
While G is not empty do
   Let v be a node of minimum degree in G
   S ← S ∪ {v}
   Remove v and its neighbors from G
end while
Output S
```

Algorithm analysis: It can show that S is an independent set of G easily. Consider the number of nodes in V/S. Each time we choose v from G to s, we delete at most d nodes in the G since each node has degree $\leq d$, that means $|V/S| \leq d\,|S|$. Notice that $|S| + |V/S| = |V|$. So $|S| \geq \frac{1}{d+1}|V| \geq \frac{1}{d+1} opt$

6. (15 points) A subsequence is **palindromic** if it is the same whether read left to right or right to left. For instance, the sequence

$$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

has many palindromic subsequences, including A, C, G, C, A and A, A, A, A. Devise an algorithm that takes a sequence $x[1, \dots, n]$, and returns the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

Sol: Let L[i][j] be the maximal length of the palindromic subsequence of x[i] to x[j]. Then L[i][j] satisfies:

$$L[i][j] = \begin{cases} 1 & \text{if } i=j \\ 2 & \text{if } j=i+1 \text{ and } x[i]=x[j] \\ \max\{L[i+1][j], L[i][j-1]\} & \text{if } x[i] \neq x[j] \\ \max\{L[i+1][j], L[i][j-1], 2+L[i+1][j+1]\} & \text{o.w.} \end{cases}$$

Then we can get the algorithm:

```
For i=1 to n
    Update L[i][i] and L[i][i+1]
For i=2 to n-1
  For j=1 to n-i
     Update L[j][i+j]
Return L[1][n]
```

It can be easily seen that the algorithm can be finished in $O(n^2)$.

7. (10 points) The **Maximum Cut** problem is defined as follows: Given an undirected graph $G = (V, E)$ along with a nonnegative weight $w_{ij} \geq 0$ for each $(i, j) \in E$. The goal is to partition the vertex set into two parts, U and $W = V - U$, so as to maximize the weight of the edges whose two endpoints are in different parts. Give a 2-approximation randomized algorithm for maximum cut problem.

Sol: The algorithm is very easy, we just need place each vertex $v \in V$ into U independently with probability $\frac{1}{2}$.

Algorithm Analysis: Consider a random variable X_{ij} that is 1 if the edge (i, j) is in the cut, and 0 o.w. . Let Z be the random variable equal to the total weight of edges in the cut, so that $Z = \sum_{(i,j)\in E} w_{ij} X_{ij}$. Let OPT denote that the optimal value of maximum cut instance. Then, as before by linearity of expectation and the definition of expectation of 0-1 random variable, we get that:

$$E[Z] = \sum_{(i,j)\in E} w_{ij} E[X_{ij}] = \sum_{(i,j)\in E} w_{ij} \Pr[Edge(i,j) \text{ in the cut}]$$

In this case, the probability that a specific edge (i,j) is in the cut is easy to calculate: since the two endpoints are placed in the sets independently, they are in different sets with probability equal to. Hence,

$$E[Z] = \frac{1}{2} \sum_{(i,j)\in E} w_{ij} \geq \frac{1}{2} OPT$$

8. (15 points) The **Weighted Vertex Cover** problem is defined as follows: Given an undirected graph $G = (V, E)$ where $|V| = n$ and $|E| = m$ and a cost function on vertices $c: V \to Q^+$, find a subset $C \subseteq V$ such that every edge $e \in E$ has at least one endpoint in C and C has a minimum cost. Use the primal-dual method to give a 2-approximation algorithm for this problem, and prove the guarantee.

Sol:

The relaxed primal LP of the problem is

$$\begin{array}{lll} \min & \sum_{i \in V} c(i) x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \mathit{for\ all}\ (i,j) \in E \\ & x_i \geq 0 & \mathit{for\ all}\ i \in V \end{array}$$

The dual is

$$\begin{array}{lll} \max & \sum_{e \in E} y_e & \\ \text{s.t.} & \sum_{e=(i,j)} y_e \leq c(i) & \mathit{for}\ i \in V \\ & y_e \geq 0 & \mathit{for\ all}\ e \in E \end{array}$$

To obtain a factor-2 algorithm using the primal-dual schema, we choose $\alpha = 1$ and $\beta = 2$. We will construct an algorithm whose output satisfies the following relaxed complementary slackness conditions:

Primal conditions:	$\forall i \in V: x_i \neq 0 \Rightarrow \sum_{e=(i,j)} y_e = c(i)$
Dual condition:	$\forall e \in E: y_e \neq 0 \Rightarrow 1 \leq x_i + x_j \leq 2$

The two sets of conditions naturally suggest the following algorithm.

Algorithm (Weighted Vertex Cover – factor 2)

$$\boxed{\sum_{e=(i,j)} y_e = c(i)}$$

1. Initialization: x ← 0; y ← 0
2. While (exists an edge (i, j) such that neither i nor j are tight):
   a) Pick such an edge e, and raise y_e until some vertex *i* or *j* goes tight.
   b) If *i* goes tight, then $x_i = 1$. Similarly, if *j* goes tight, then $x_j = 1$.
3. $C \leftarrow \{i \in V: x_i = 1\}$
4. Output the vertex cover *C*.

Proof:

1. *x* is a feasible primal solution and *y* is a feasible dual solution.

   Primal: When the while loop terminates, in every edge (i, j), either i or j are tight. That is, either $x_i = 1$ or $x_j = 1$.

   Dual: After a vertex goes tight, all edges connecting it won't be raised.

2. The relaxed complementary slackness conditions hold for *x* and *y*.

   Primal conditions: when $x_i = 1$, then it is tight.

Dual conditions: Each x_i can at most be 1.

3. The approximation factor is 2.

The approximation factor is $\alpha\beta = 2$ by proposition 15.1 in *Approximation Algorithms.* http://basics.sjtu.edu.cn/~liguoqiang/teaching/Galgo17/books/approximationfull.pdf