Advanced Algorithm: Assignment #2

Due on Dec 7, 2017 at 3:10pm

Professor Yitong Yin

Hao Li 141070027

Problem 1

Consider the following optimization problem.
Instance: n positive intergers $x_1 < x_2 < \cdots < x_n$.
Find two disjointed nonempty subsets $A, B \subset \{1, 2, \ldots, n\}$ with $\sum_{i\in A} x_i \geq \sum_{i\in B} x_i$, such that the ratio $\frac{\sum_{i\in A} x_i}{\sum_{i\in B} x_i}$ is minimized.
Give a pseudo-polynomial time algorithm for the problem, and then give an FPTAS for the problem based on the pseudo-polynomial time algorithm.
Solution
part 1
the problem is to find two disjoint nonempty subsets, a heuristic strategy is for a sum of some element in a set, and call the set C_1, then find the maximum sum of set C_2 such that $C_1 \cap C_2 = \emptyset$. and we can use dynamic programming to solve the problem.
Here is the algorithm:
the main idea of the algorithm is: for a sum, if we can find 2 subsets and the two subsets are disjointed and

Algorithm 1: Dynamic programming algorithm to find two subsets

input :
A set $\{x_1 < x_2 < \cdots < x_n\}$ with positive intergers;
output:
two disjoint nonempty subsets $A, B \subset \{1, 2, \ldots, n\}$ with $\sum_{i\in A} x_i \geq \sum_{i\in B} x_i$, such that the ratio $\frac{\sum_{i\in A} x_i}{\sum_{i\in B} x_i}$ is minimized.;

initialize table $(n+1) \times (1 + sum_{i=1}^{n} a_i)\{0, \emptyset\}$, the element of the table is a struct with int a, set b. $a \in \{0, 1\}, b \subset \{1, 2, \ldots, n\}$. table[0,0].a=1, table[0,0].b=$\emptyset$.;
for *i=1 to n* **do**
  table[i,0].a=0;
  table[i,0].b=$\emptyset$;
end
for *j=1 to* $sum_{i=1}^{n} a_i$ **do**
  table[0,j].a=0;
  table[0,j].b=$\emptyset$;
end
for *j=1 to* $sum_{i=1}^{n} a_i$ **do**
  for *i=i to n* **do**
    if $j \geq x_i$ *and* $\exists k \in \{0, \ldots, i-1\} with table[k, j - x_i] = 1$ **then**
      $table[i, j].a = 1$;
      $table[i, j].b = table[k, j - a_i] \cup \{i\}$;
    end
    else
      table[i,j].a=0;
      table[i,j].b=$\emptyset$;
    end
  end
  if $i_1 \neq i_2$ *and* $table[i_1, j].a = table[i_2, j].a$ **then**
    Stop;
  end
end

Problem 1 continued on next page. . .

we can find a optimal instance of 1. and if the optimal is not 1, the for every sum, find a sum less than the sum and the two sets are disjointed and get the minimum.
we can take two sets as examples:
for set {1,2,3,4}:

	0	1	2	3	4	
0	1	0	0	0	0	
1	0	1	0	0	0	
2	0	0	1	0	0	
3	0	0	1	1	0	stop
4	0					
5	0					
6	0					
7	0					
8	0					
9	0					
10	0					

Figure 1: an example

and we can find we can get a sum 3, and make the fraction is 1.
another example:
for set {1,2,4,8}:

	0	1	2	4	8
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	1	0
6	0	0	0	1	0
7	0	0	0	1	0
8	0	0	0	0	1
9	0	0	0	0	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	1
13	0	0	0	0	1
14	0	0	0	0	1
15	0	0	0	0	1

Figure 2: an example

and we can find the optional subset sum couple (14,1),(13,2),(12,3),(11,4),(10,5),(9,6),(8,7),(6,1),(5,2),(4,3),(2,1).
and we can get the optimal (8,7).
the we can prove why the algorithm is optimal:
if the optimum of the instance is 1, then the algorithm can always get 1. if the optimum of the instance is not 1, then we get the couple set (sol_a, sol_b), if there exists a couple set(opt_a, opt_b), such that $\frac{opt_a}{opt_b} < \frac{sol_a}{sol_b}$, but we have:

$$\frac{sol_a}{sol_b} < \frac{opt_a}{opt_{sol_a}} < \frac{opt_a}{opt_b}$$

so the algorithm can always achieve the optimum.
and obviously the time complexity is $O(n \times sum_{i=1}^{n} a_i)$.

fptas

we can find why the dynamic programming is pseudo-polynomial, because the sum of all the set number is

not depend on n. so we can scale the sum into some polynomial number.
we find a function:

$$k(m) = \tfrac{\epsilon^2 \times x_m}{2 \times m}$$

then find the greatest number n_0 such that $k(n_0) < 1$, then we find the sum of the set number of $\{x_1, x_2, \dots, x_{n_0}\}$ is polynomial by n.

Claim 1. *we can take polynomial time to find the optimal ratio of set $\{x_1, x_2, \dots, x_m\}$, if $m \leq n_0$*

Proof.

$$x_{n_0} < \frac{2n}{\epsilon^2} \quad \textit{given } k(n_0) < 1$$

and we have $x_1 < x_2 < \dots < x_{n_o}$, so the sum $S < \frac{2n^2}{\epsilon^2}$, then using the pseudo-polynomial algorithm, the time complexity is polynomial. □

then we need to look for a way to solve $m > n_0$.
we find a function:

$$x_i^{'} = \left\lfloor \frac{x_i}{k(m)} \right\rfloor, \ \textit{for } i = 1, 2, \dots, m$$

then $x_m^{'} = \lfloor \frac{2m}{\epsilon^2} \rfloor$. and we wanna to find all the $x_i^{'}$ that $x_i^{'} > \frac{m}{\epsilon}$. Assume there are t numbers, which is $x_{m-t+1}^{'}, \dots, x_m^{'}$. since $\epsilon \leq 1$, so we have $x_m^{'} \frac{m}{\epsilon}$, so we get $t \geq 1$. then we can distinguish the situation into 2 cases by the value of t.

condition 1:

t=1, this is a expected condition, let j be the smallest noninterger such that $x_{j+1} + \dots + x_{m-1} < x_m$, then the solution will be $S_1 = \{m\}$ and $S_2 = \{j, j+1, \dots, m-1\}$, and if j=0, and $S_1 = \{0, 1, \dots, m-1\}$, $S_2 = \{m\}$.

condition 2:

$t > 1$, if we use the pseudo-polynomial algorithm and we will take only polynomial time on the condition.

Claim 2. *we can take polynomial time to find the optimal ratio of set $\{x_1^{'}, x_2^{'}, \dots, x_m^{'}\}$*

Proof. because we have another scale and make $x_m^{'} < \frac{2m}{\epsilon^2}$, so the sum is less than $\frac{2n^2}{\epsilon^2}$. □

condition2.1:

if the optimum of the set $\{x_1^{'}, \dots, x_m^{'}\}$ is 1, on this condition, the algorithm can return the solution which realize the optimum for the sets.
then we call the solution two set S_1 and S_2.

condition2.2:

if the optimum of the set $\{x_1^{'}, \dots, x_m^{'}\}$, which is denoted by $opt(I_m^{'})$, is more than 1. and we can use a trick to get the two sets.
let $I_m^{'}$ denotes the set of $x_i^{'}$ which is greater than $\frac{m}{\epsilon}$. we denote S_1, S_2 is the set and $S_1, S_2 \subset I_m^{'}$, and S_1, S_2 is disjointed. then we have 3^{t-1} pairs:

$$x_{m-t+i} \in S_1 \ \textit{and} \ \ x_{m-t+i} \notin S_2$$
$$x_{m-t+i} \in S_2 \ \textit{and} \ \ x_{m-t+i} \notin S_1$$
$$x_{m-t+i} \notin S_1 \ \textit{and} \ \ x_{m-t+i} \notin S_2$$

for $1 < i < t-1$, obviously $x_m \in S_1$ and $x_m \in S_2$. and for every pair, define:

$$R_1 = \begin{cases} S_1 & \sum_{i\in S_1} x_i > \sum_{i\in S_2} x_i \\ S_2 & \sum_{i\in S_1} x_i > \sum_{i\in S_2} x_i \end{cases}$$

and R_2 is the other set. let j be the smallest nonnegative integer such that: $x_j + \cdots + x_m - t + R_2 < R_1$. and we make the two set, $SET_1 = R_1$, and $SET_2 = R_2 \cup \{x_j + \cdots + x_m - t\}$. and we choose from the 3^{t-1} pairs to find a smallest ratio.
then we can solve all condition in polynomial time.
Next we prove the above algorithm achieve $a(1+\epsilon) - approximation$:

Proof. $x_m^{'} \leq \frac{2m}{\epsilon^2}, then we have \sum_{i=m-t+1}^{m} x_i^{'} < \frac{2m^2}{\epsilon^2}$, therefore $2^t \leq \frac{2m^2}{\epsilon^2}$, then $t \leq 2\log(\frac{m}{\epsilon}+1)$.
if $m \leq n_0$, which is discussed before, we can get the optimum ratio. therefore, we talked $m > n_0$:
in condition 1: if j=0, then the given solution is optimum, which is obvious. and if j¿0, then:

$$\frac{\sum_{i\in S_1} x_i}{\sum_{i\in S_2} x_i} \leq 1 + \frac{x_j}{x_m} < l + \epsilon. \tag{1}$$

this equation is obviously, becase x_j is small compared to x_m. □

in fact, we can have the same comparison in condition 2, only the comparison is more complex.
in condition 2.1, we have:

$$\frac{\sum_{i\in S_1} x_i}{\sum_{i\in S_2} x_i} \leq \frac{\sum_{i\in S_1} k(m)\times(1+x_i^{'})}{\sum_{i\in S_2} k(m)\times x_i^{'}} = 1 + \frac{\|S_1\|}{\sum_{i\in S_2} x_i^{'}} \leq 1 + \frac{t}{m/\epsilon} < 1+\epsilon \tag{2}$$

in condition 2.2, if j=0, which is similar to condition 1. then the given solution is optimal. and if j¿0, we have:

$$\sum_{i\in R_2} x_i + \sum_{i=j+1}^{m-t} x_i < \sum_{i\in R_1} x_i \leq \sum_{i\in R_2} x_i + \sum_{i=j}^{m-t} x_j. \tag{3}$$

since j is the critical point. so we have:

$$\frac{\sum_{i\in S_1} x_i}{\sum_{i\in S_2} x_i} \leq 1 + \frac{x_j}{x_m} < 1+\epsilon \tag{4}$$

which is similar to condition 1. and we prove the approximation ratio.

Problem 2

In the maximum directed cut (MAX-DICUT) problem, we are given as input a directed graph$G(V, E)$.The goal is to partition Vinto disjoint$S$and T so that the number of edges in $E(S, T) = \{(u, v) \in E \mid u \in S, v \in T\}$is maximized. The following is the integer program for MAX-DICUT:

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,v)\in E} y_{u,v} \\ \text{subject to} \quad & y_{u,v} \le x_u, && \forall (u,v) \in E, \\ & y_{u,v} \le 1 - x_v, && \forall (u,v) \in E, \\ & x_v \in \{0,1\}, && \forall v \in V, \\ & y_{u,v} \in \{0,1\}, && \forall (u,v) \in E. \end{aligned}$$

Let $x_v^*, y_{u,v}^*$denote the optimal solution to the LP-relaxation of the above integer program.

- Let$x_v^*, y_{u,v}^*$ denote the optimal solution to the LP-relaxation of the above integer program.
- Apply another randomized rounding such that for every $v \in V$, $\hat{x}_v = 1$ independently with probability$1/4 + x_v^*/2$. Analyze the approximation ratio for this algorithm.

Solution
part 1: when we apply the rounding:

$$\hat{x}_v = \begin{cases} 1 & \textit{with possibility } x_v^* \\ 0 & \textit{with possibility } 1 - x_v^* \end{cases}$$

we have:

$$OPT < OPT_L P = \sum_{uv \in E} y_{u,v}^* \tag{5}$$

and we also have:

$$SOL = \sum_{uv \in E} \hat{y}_{u,v} \tag{6}$$

and we also have:

$$\hat{y}_{u,v} = \begin{cases} 1 & \hat{x}_v = 0 \wedge \hat{x}_u = 1 \\ 0 & \hat{x}_v = 1 \vee \hat{x}_u = 0 \end{cases}$$

then we get:

$$SOL = \sum_{uv \in E} \hat{y}_{u,v} = \sum_{uv \in E} (1 - x_v^*) \times (x_u^*) \ge \sum_{uv \in E} (y_{u,v}^*)^2 \tag{7}$$

by arithmetic-geometric mean inequality, we have:

$$SOL \ge \sum_{uv \in E} (y_{u,v}^*)^2 \ge \frac{(\sum_{uv \in E} y_{u,v}^*)^2}{\|E\|} \ge \frac{\sum_{uv \in E} y_{u,v}^*}{2} \ge \frac{OPT}{2} \tag{8}$$

then we can get the approximation ratio is $\frac{1}{2}$.
part 2
this part is similar to the part above, but much easier. when we apply the rounding:

$$\hat{x}_v = \begin{cases} 1 & \textit{with possibility } \frac{1}{4} + \frac{x_v^*}{2} \\ 0 & \textit{otherwise} \end{cases}$$

we have:

$$OPT < OPT_L P = \sum_{uv \in E} y^*_{u,v} \tag{9}$$

and we also have:

$$SOL = \sum_{uv \in E} \hat{y}_{u,v} \tag{10}$$

and we also have:

$$\hat{y}_{u,v} = \begin{cases} 1 & \hat{x}_v = 0 \wedge \hat{x}_u = 1 \\ 0 & \hat{x}_v = 1 \vee \hat{x}_u = 0 \end{cases}$$

then we get:

$$SOL = \sum_{uv \in E} \hat{y}_{u,v} = \sum_{uv \in E} (\frac{1}{4} + \frac{x^*_u}{2}) \times (\frac{3}{4} - \frac{x^*_v}{2}) \tag{11}$$

and by the meaning of the problem, we also have:

$$\begin{aligned} x^*_u &\geq y^*_{u,v} \\ 1 - x^*_v &\geq y^*_{u,v} \end{aligned}$$

then we have:

$$\begin{aligned} SOL &= \sum_{uv \in E} (\frac{1}{4} + \frac{x^*_u}{2}) \times (\frac{3}{4} - \frac{x^*_v}{2}) \\ &\geq \sum_{uv \in E} (\frac{1}{4} + \frac{y^*_{u,v}}{2}) \times (\frac{1}{4} + \frac{y^*_{u,v}}{2}) \\ &= \sum_{uv \in E} (\frac{1}{4} + \frac{y^*_{u,v}}{2}) + \frac{y^*_{u,v}}{2} \\ &\geq \sum_{uv \in E} (\frac{y^*_{u,v}}{2}) \\ &\geq \frac{\sum_{uv \in E} y^*_{u,v}}{2} \\ &= \frac{OPT_{LP}}{2} \\ &\geq \frac{OPT}{2} \end{aligned}$$

so we get the approximation ratio $\frac{1}{2}$.

Problem 3

Recall the MAX-SAT problem and its integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} y_j \\ \text{subject to} \quad & \sum_{i \in S_j^+} x_i + \sum_{i \in S_j^-} (1 - x_i) \geq y_j, && 1 \leq j \leq m, \\ & x_i \in \{0, 1\}, && 1 \leq i \leq n, \\ & y_j \in \{0, 1\}, && 1 \leq j \leq m. \end{aligned}$$

Recall that$S_j^+, S_j^- \subseteq \{1, 2, \ldots, n\}$ are the respective sets of variables appearing positively and negatively in clause j.
Let x_i^*, y_j^*denote the optimal solution to the LP-relaxation of the above integer program. In our class we learnt that if $\hat{x}_i$ is round to 1 independently with probability x_i^*, we have approximation ratio $1 - 1/\text{e}$.
We consider a generalized rounding scheme such that every $\hat{x}_i$ is round to 1 independently with probability $f(x_i^*)$ for some function $f : [0, 1] \to [0, 1]$ to be specified.

- Suppose $f(x)$ is an arbitrary function satisfying that $1 - 4^{-x} \leq f(x) \leq 4^{x-1}$ for any $x \in [0, 1]$. Show that with this rounding scheme, the approximation ratio (between the expected number of satisfied clauses and OPT) is at least 3/4.
- Derandomize this algorithm through conditional expectation and give a deterministic polynomial time algorithm with approximation ratio 3/4.
- Is it possible that for some more clever f we can do better than this? Try to justify your argument.

Solution
part 1
in the class, we learn a $1 - \frac{1}{e}$-approximation round. the core equation of the approximation is:

$$Pr[C_j \ is \ satisfied] = 1 - \prod_{i \in S_j^+} (1 - x_i^*) \times \prod_{i \in S_j^-} x_i^* \geq (1 - \frac{1}{e}) y_j^* \tag{12}$$

and for this problem, we want to prove:

$$Pr[C_j \ is \ satisfied] = 1 - \prod_{i \in S_j^+} (1 - f(x_i^*)) \times \prod_{i \in S_j^-} f(x_i^*) \geq \frac{3}{4} y_j^* \tag{13}$$

that is what I need to prove.

Claim 3. *a function $f : [0, 1] \to [0, 1]$ has approximation $\frac{3}{4}$ if:*

$$1 - \prod_{i=1}^{l} (1 - f(x_i^*)) \times \prod_{i=l+1}^{k} f(x_i^*) \geq \frac{3}{4} \times \min(1, \sum_{i=1}^{l} x_i^* + \sum_{i=l+1}^{k} (1 - x_i^*)) \tag{14}$$

Proof. first, the same as the class teached, we have:

$$\sum_{i \in S_j^+} x_i^* + \sum_{i \in S_j^-} (1 - x_i^*) > y_j^* \tag{15}$$

without loss of generality, we can assume all variables in the clause are unnegated. Indeed, if one clause appear negative in clause C_j, we can replace if by its negation. so we can have:

$$\sum_{i=1}^{l} x_i^* + \sum_{i=l+1}^{k} (1 - x_i^*) > y_j^* \tag{16}$$

therefore, we have:

$$1 - \prod_{i \in S_j^+} (1 - f(x_i^*)) \times \prod_{i \in S_j^-} f(x_i^*) \geq \frac{3}{4} y_j^* \tag{17}$$

easy to prove. □

Claim 4. *a function $g : [0,1] \to [0,1]$ be a function satisfying:*

$$1 - \prod_{i=1}^{k} (1 - g(x_i^*)) \geq \frac{3}{4} \times \min(1, \sum_{i=1}^{k} x_i^*) \tag{18}$$

for all k and all x.
function $f : [0,1] \to [0,1]$: satisfying:

$$g(x) \leq f(x) \leq 1 - g(1-x) \tag{19}$$

then f has approximation $\frac{3}{4}$.

Proof. we have:

$$\begin{aligned}
& 1 - \prod_{i=1}^{l} (1 - f(x_i^*)) \times \prod_{i=l+1}^{k} f(x_i^*) \\
& \geq 1 - \prod_{i=1}^{l} (1 - g(x_i^*)) \times \prod_{i=l+1}^{k} (1 - g(1 - x_i^*)) && \textit{using } g(x) \leq f(x) \leq 1 - g(1-x) \\
& \geq \frac{3}{4} \times \min(1, \sum_{i=1}^{l} x_i^* + \sum_{i=l+1}^{k} (1 - x_i^*)) && \textit{using arithmetic} - \textit{geometric mean inequality}
\end{aligned}$$

□

then if we can prove that:

$$g(x) = 1 - 4^{-x} \tag{20}$$

satisfy:

$$1 - \prod_{i=1}^{k} (1 - g(x_i^*)) \geq \frac{3}{4} \times \min(1, \sum_{i=1}^{k} x_i^*) \tag{21}$$

then we have:

$$\begin{aligned}
1 - \prod_{i=1}^{k} (1 - g(x_i^*)) &= 1 - \prod_{i=1}^{k} (4^{-x}) \\
&= 1 - 4^{-\sum_{i=1}^{k} x_i^*} \\
&= 1 - 4^{-X} \\
&= g(X)
\end{aligned}$$

and $X = \sum_{i=1}^{k} x_i^*$. we only need to prove $g(X) \geq \frac{3}{4} \times X$, for any $X \in [0,1]$. and we can use Derivative to get the solution.
then we prove it.
part 2:
the algorithm is given in the Vazirani's "Approximation Algorithms" book.

Algorithm 2: Derandomized algorithm for MAX-SAT problem

input :
a set of clauses $C_1, C_2, \ldots, C_m$;
output:
an assignment $x \in \{true, false\}^n$ that maximizing the satisfied clauses;

use the derandomized factor $\frac{1}{2}$ algorithm to get a truth assignment;
use the derandomized factor $1 - \frac{1}{e}$ algorithm to get a truth assignment;
return *the better of the two assignments.*

part 3:
I think it is impossible to find some more clever f do better than $\frac{3}{4}$.

Claim 5. *the relative gap between IP and LP is $\frac{3}{4}$, which is:*

$$Y_{LP}^* \leq \frac{4}{3} Y_{IP}^* \tag{22}$$

Proof. we have get the $\frac{3}{4}$ approximation ratio. so that the gap $\geq \frac{3}{4}$.
consider a 2SAT instance:

$$\begin{cases} x_1 \vee x_2 \\ x_1 \vee \bar{x}_2 \\ \bar{x}_1 \vee x_2 \\ \bar{x}_1 \vee \bar{x}_2 \end{cases}$$

the LP solution $x_i = \frac{1}{2}$ for all i and $y_j = 1$ for all j is optimal for any instance without unit clause, and we find Z_{IP}^* is 3.
then we get a counterexample if the gap is above $\frac{3}{4}$. so it is impossible to find some more clever f do better than $\frac{3}{4}$. $\square$

Problem 4

The following is the weighted version of set cover problem: Given m subsets $S_1, S_2, \ldots, S_m \subseteq U$, where U is a universe of size $n = |U|$, and each subset S_i is assigned a positive weight $w_i > 0$, the goal is to find a $C \subseteq \{1, 2, \ldots, m\}$ such that $U = \bigcup_{i \in C} S_i$ and the total weight $\sum_{I \in C} w_i$ is minimized.

- Give an integer program for the problem and its LP relaxation.
- Consider the following idea of randomized rounding: independently round each fractional value to $\{0, 1\}$ with the probability of the fractional value itself; and repeatedly apply this process to the variables rounded to 0 in previous iterations until U is fully covered. Show that this can return a set cover with $O(\log n)$ approximation ratio with probability at least 0.99.

solution
part 1 easy.
the interger program is:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} x_i \times w_i \\ \text{subject to} \quad & \sum_{i=1, e \in S_i}^{m} x_i \geq 1, \quad e \in U, \\ & x_i \in \{0, 1\}, \qquad 1 \leq i \leq m \end{aligned}$$

and its LP relaxation is :

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} x_i \times w_i \\ \text{subject to} \quad & \sum_{i=1, e \in S_i}^{m} x_i \geq 1, \quad e \in U, \\ & x_i \in [0, 1], \qquad 1 \leq i \leq m \end{aligned}$$

part 2 as shown in the problem, the algorithm is running as:

the problem is difficult to prove that the approximation ratio is $O(\log n)$, so I transform the idea of the problem.
we wanna to prove that there is a probability at most 0.99 that the loop has been executed for $O(\log n)$ times.(the while state)
if we prove that, we have:

$$\begin{aligned} & SOL_i < OPT \\ & SOL = SOL_1 + SOL_2 + \cdots + SOL_k < O(\log n) \times OPT \end{aligned}$$

Claim 6. *there is a probability that at most e^{-k} that the while loop is executed for more than $\ln \|U\| + k$ times.*

Algorithm 3: Randompick algorithm for the weighted version problem

input :

m subsets $S_1, S_2, ...S_m \subset U$;

size n;

a weight function w→ R :

output:

the minimum size of $\|C\|$ that $C \subset \{1, 2, \dots, m\}$;

```
initialize C = ∅
initialize U=S_1 ⋃ S_2 ⋃ ··· ⋃ S_n;
while U≠ ∅ do
    for i∉ C do
        with possibility ||S_i||/||U||, set x_i=1.;
        if x_i=1 then
            C = C ⋃{i};
            U = U − S_i
        end
    end
end
return C
```

Proof. the probability that we need to run the while loop for more than $\ln \|U\| + k$ times is the same as the probability that if we run the body of the while loop for exactly $\ln \|U\| + k$ steps, we end up with some uncovered elements.

Consider an element $u \in U$. For each iteration of while loop, there is a probability at most $\frac{1}{e}$ that u is not covered by the sets added to C in that iteration. the probability that u is not covered after $\ln \|U\| + k$ iterations is then at most:

$$(\frac{1}{e})^{\ln \|U\|+k} = \frac{1}{\|U\|} \times e^{-k} \tag{23}$$

the probability that, after $\ln \|U\| + k$ iterations, there is an element that is not covered, is at most the sum over all u of the probability that u is not covered, which is at most e^{-k}. □

then we can get the 0.99. just find a k to make $e^{-k} = 0.01$. it is easy.

Problem 5

Recall that the instance of set cover problem is a collection of m subsets $S_1, S_2, \ldots, S_m \subseteq U$, where U is a universe of size $n = |U|$. The goal is to find the smallest $C \subseteq \{1, 2, \ldots, m\}$ such that $U = \bigcup_{i \in C} S_i$. The frequency f is defined to be $\max_{x \in U} |\{i \mid x \in S_i\}|$.

- Give the primal integer program for set cover, its LP-relaxation and the dual LP.
- Describe the complementary slackness conditions for the problem.
- Give a primal-dual algorithm for the problem. Present the algorithm in the language of primal-dual scheme (alternatively raising variables for the LPs). Analyze the approximation ratio in terms of the frequency f.

part 1:
Also we introduce x_i for every set S_i with intended meaning that $x_i = 1$ means that S_i is selected, while 0 otherwise. then we can express the set cover problem as the following integer linear program:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} x_i \\
\text{subject to} \quad & \sum_{i=1, e \in S_i}^{m} x_i \geq 1, \quad e \in U, \\
& x_i \in \{0, 1\}, \qquad 1 \leq i \leq m
\end{aligned}
$$

then we have its LP-relaxation:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} x_i \\
\text{subject to} \quad & \sum_{i=1, e \in S_i}^{m} x_i \geq 1, \quad e \in U, \\
& x_i \geq 0, \qquad 1 \leq i \leq m
\end{aligned}
$$

the last inequation may be $x_i \in [0, 1]$, but obviously if $x_i = 1$ can satisfy all the need, so we don't need to make $x_i > 1$ when finding the optimal result. so it is unnecessary to restrict $x_i < 1$.
then for every $e \in U$, we introduce a parameter y_e, then we get its dual LP, which is:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in U} y_e \\
\text{subject to} \quad & \sum_{e \in S_i} y_e \leq 1, \quad i \in \{1, 2, \ldots, m\}, \\
& y_e \geq 0, \qquad e \in U
\end{aligned}
$$

part 2:
the primal complementary slackness conditions is:
primal complementary slackness conditions. for each S_i:

$$\textit{either } x_i = 0 \textit{ or } \sum_{e \in S_i} y_e = 1$$

the condition states that: "pick only tight sets into the cover."
the dual complementary slackness conditions is:
dual complementary slackness conditions. For each $e \in U$:

$$either\ y_e = 0\ or\ \sum_{e \in S_i} x_i \leq f.$$

part 3:
then we want to prove that the algorithm is a $f - approximation$ algorithm for SET COVER.

Algorithm 4: Primal-dual algorithm for set cover problem.

input :
m subsets $S_1, S_2, ...S_m U$;
size n;
output:
Vector $x \in \{0, 1\}^k$;

initialize x=0,y=0
initialize U=$S_1 \bigcup S_2 \bigcup \cdots \bigcup S_n$;
while $U \neq \emptyset$ **do**
pick an uncovered element, say e, and raise y_e until some set goes tight;
pick all tight sets S in the cover, and set corresponding $x = 1$;
declare all the elements occuring in these sets are covered;
end
return x

Claim 7. *the primal-dual algorithm for set cover problem is an $f - approximation$ algorithm.*

Proof. at the end of the algorithm, there will be no uncovered elements. depend on the slackness conditions and we can obtain:

$$\begin{aligned}
\sum_{i=1}^{m} x_i &= \sum_{i=1}^{m} \sum_{e \in S_i} y_e \times x_i \\
&= \sum_{i=1}^{m} \sum_{e \in S_i} x_i \times y_e \\
&\leq \sum_{i=1}^{m} f \times y_e \\
&= f \times \sum_{i=1}^{m} y_e \\
&= f \times \sum_{e \in U} y_e
\end{aligned}$$

Problem 5 continued on next page. . .

the deduction uses the inquation:

$$\sum_{e \in S_i} x_i \leq f$$

$$\sum_{e \in S_i} y_e = 1$$

and $\sum_{i=1}^{m} y_e$ and $\sum_{e \in U} y_e$ just different expression of the same things. $\square$