

Problem Set 12:

潘光之 181240045

Problem 1.

10

Allocate two array $M[0 \dots n]$ and $N[0 \dots n]$. For any integer $0 \leq i \leq j \leq n$, $M[i, j](N[i, j])$ represents whether $s[i \dots j]$ can be parenthesized so that the resulting value is T(F).

We have the following recursion:

$$M[i, j] = \begin{cases} S[2i], i=j \\ \bigvee_{i \leq k < j} (M[i, k] \wedge M[k+1, j]), S[2k+1] = ' \wedge ' \\ \bigvee_{i \leq k < j} (M[i, k] \vee M[k+1, j]), S[2k+1] = ' \vee ' \\ \bigvee_{i \leq k < j} [(M[i, k] \wedge N[k+1, j]) \vee (N[i, k] \wedge M[k+1, j])], S[2k+1] = ' \oplus ' \end{cases}$$

$$N[i, j] = \begin{cases} \sim S[2i], i=j \\ \bigvee_{i \leq k < j} (N[i, k] \vee N[k+1, j]), S[2k+1] = ' \wedge ' \\ \bigvee_{i \leq k < j} (N[i, k] \wedge N[k+1, j]), S[2k+1] = ' \vee ' \\ \bigvee_{i \leq k < j} [(M[i, k] \wedge M[k+1, j]) \vee (N[i, k] \wedge N[k+1, j])], S[2k+1] = ' \oplus ' \end{cases}$$

Alg. for $i=0$ to n :

$$M[i, i] = S[2i]$$

$$N[i, i] = S[2i]$$

for $l=1$ to n :

for $i=0$ to $n-l$:

$$j=i+l$$

$$M[i, j] = F$$

$$N[i, j] = F$$

for $k=i$ to $j-1$:

if $S[2k+1] == ' \wedge '$:

$$M[i, j] = M[i, j] \vee (M[i, k] \wedge M[k+1, j])$$

$$N[i, j] = N[i, j] \vee (N[i, k] \wedge M[k+1, j])$$

else if $S[2k+1] == ' \vee '$:

$$M[i, j] = M[i, j] \vee (M[i, k] \vee M[k+1, j])$$

$$N[i, j] = N[i, j] \vee (N[i, k] \wedge N[k+1, j])$$

else:

$$M[i, j] = M[i, j] \vee [(M[i, k] \wedge N[k+1, j]) \vee (N[i, k] \wedge M[k+1, j])]$$

$$N[i, j] = N[i, j] \vee [(M[i, k] \wedge M[k+1, j]) \vee (N[i, k] \wedge N[k+1, j])]$$

return $M[0, n]$

Problem 2:

10

分而两个数组 $\text{MAX}[1 \dots n]$, $\text{MIN}[1 \dots n]$, 其中 $\text{MAX}[i]$ 表示以 $A[i]$ 为右端点的 subarray 的最大 product, $\text{MIN}[i]$ 表示以 $A[i]$ 为右端点的 subarray 的最小 product. ($1 \leq i \leq n$)

Alg. $\text{res} = \max\{1, A[1]\}$

$\text{MAX}[1] = A[1]$

$\text{MIN}[1] = A[1]$

for $i=2$ to n :

if $A[i] > 0$:

$$\text{MAX}[i] = \max\{\text{MAX}[i-1] * A[i], A[i]\}$$

$$\text{MIN}[i] = \min\{\text{MIN}[i-1] * A[i], A[i]\}$$

██████████

else:

$$\text{MAX}[i] = \max\{\text{MIN}[i-1] * A[i], A[i]\}$$

$$\text{MIN}[i] = \min\{\text{MAX}[i-1] * A[i], A[i]\}$$

$$\text{res} = \max\{\text{MAX}[i], \text{res}\}$$

return res

Problem 3:

把所有点按 x -coordinate 顺序排序, 记排序后的 points 为 P_1, P_2, \dots, P_n , 显然, P_1, P_n 分别是 leftmost point 和 rightmost point. 记 $P(i, j)$ 为如下路径的长度 + ($i \leq j$) 最短

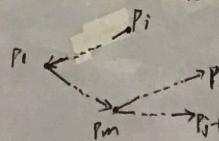
start at point P_i , go strictly leftward to the leftmost point P_1 , and then go strictly rightward to the right point P_j , where all points P_1, \dots, P_j should be included and no point can be passed twice (except P_i in $P(i, i)$).

1° 当 $i=1$ 时, $P(i, j)$ 的一个端点为 P_1 , 特别地, 当 $j=2$ 时, $P(i, j) = \text{dist}(P_i, P_2)$.

2° $i < j-1$ 时. 首先证明边 (P_{j-1}, P_j) 必在 $P(i, j)$ 中: 从 P_i 开始

由于 $P(i, j)$ 的定义且 $i < j-1$, 点 P_{j-1}, P_j 不可能在严格向左的路径中被经过。假设设 (P_{j-1}, P_j) 不在 $P(i, j)$ 中, 设 rightward path 中存在路径 $P_m \rightarrow P_{j-1}$ 和 $P_m \rightarrow P_j$, $m < j$. 对 $\text{length}(P_m \rightarrow P_{j-1})$

会发现如下情况。



10

不满足 closed path 性质, 矛盾。

$$P(i, j) = P(i, j-1) + \text{dist}(P_{j-1}, P_j)$$

3. $i=j-1$, ~~连接~~ 连接 P_j 的边有多种可能: (P_k, P_j) , $1 \leq k < i$

$$\Rightarrow P(i, j) = \min_{1 \leq k < i} \{P(k, i) + \text{dist}(P_k, P_j)\}$$

4. $i=j$. 易证, 边 (P_{j-1}, P_j) 必在 $P(i, j)$ 中.

证: If P_{j-1} 在 leftward 过程中经过 $\Rightarrow (P_{j-1}, P_j)$ 在 $P(i, j)$ 中

If P_{j-1} 在 rightward 过程中经过 \Rightarrow 证明同上。

$$\Rightarrow P(i, j) = P(i-1, i) + \cancel{\text{dist}}(P_{j-1}, P_i)$$

So the recursion is:

$$P(i, j) = \begin{cases} \text{dist}(P_i, P_j), & i=1, j=2 \\ P(i, j-1) + \text{dist}(P_{j-1}, P_j), & i < j-1 \\ \min_{1 \leq k < i} \{P(k, i) + \text{dist}(P_k, P_j)\}, & i=j-1 \\ P(i-1, i) + \text{dist}(P_{j-1}, P_i), & 2 \leq i = j \leq n \end{cases}$$

Alg.

Input: An array representing n points: $A[1 \dots n]$

Output:

Sort(A)

$$P[1, 2] = \text{dist}(A[1], A[2])$$

for $j=3$ to n :

$$P[1, j] = P[1, j-1] + \text{dist}(A[j-1], A[j])$$

$E[1, j] = (1, j-1)$ // 保存 $P(i, j)$ 的子问题, 这里显然是 $P[1, j-1]$

for $i=2$ to $n-1$:

$$P[i, i+1] = \infty$$

for $k=1$ to $i-1$:

$$\text{if } P[i, i+1] > P[k, i] + \text{dist}(A[k], A[i+1]),$$

$$P[i, i+1] = P[k, i] + \text{dist}(A[k], A[i+1])$$

$E[i, i+1] = (k, i)$ // 保存 $P(i, i+1)$ 的子问题序号 $P(k, i)$

for $j=i+2$ to n :

$$P[i, j] = P[i, j-1] + \text{dist}(A[j-1], A[j])$$

$E[i, j] = (i, j-1)$ // 保存 $P(i, j)$ 的子问题序号

$$P[n, n] = P[n-1, n] + \text{dist}(A[n-1], A[n])$$

$$E[n, n] = (n-1, n)$$

需要打印所有边 (以下代码用点的序号代替点).

```

print (n-1, n)
i=n-1
j=n
while !(i=1 and j=2):
    if i+1=j:
        k=E[i,j][1] //表示保存在E[i,j]位置的元组的第一个元素
        print (k,j)
    else:
        print (j-1, j)
    i=E[i,j][1]
    j=E[i,j][2]
print (1,2)

```

Problem 4:

(a) 设数组 $A[1 \dots n]$, where $A[i]$ represents the weight of the i^{th} item.

$\text{OPT}[i, w]$: 从 $A[1 \dots i]$ 中挑出的不超过 w 的最大重量, $1 \leq i \leq n$; $0 \leq w \leq W$

目标: $\text{OPT}[n, W]$

显然, 有如下 recursion:

$$\text{OPT}[i, w] = \begin{cases} \max\{A[i] + \text{OPT}[i-1, w-A[i]], \text{OPT}[i-1, w]\}, & \text{otherwise} \\ 0, & w=0 \\ A[i], & i=1 \text{ and } A[i] \leq w \\ 0, & i=1 \text{ and } A[i] > w \end{cases}$$

$$\text{OPT}[i, w] = \begin{cases} A[i], & i=1 \text{ and } A[i] \leq w \\ 0, & i=1 \text{ and } A[i] > w \\ \text{OPT}[i-1, w], & A[i] > w \\ 0, & w=0 \\ \max\{A[i] + \text{OPT}[i-1, w-A[i]], \text{OPT}[i-1, w]\}, & \text{otherwise} \end{cases}$$

Alg: for $w=0$ to W .

```

if A[i] ≤ w:
    OPT[i, w] = A[i]
else:
    OPT[i, w] = 0

```



```

for i=2 to n:
    OPT[i, 0] = 0
    for w=1 to W:
        if A[i] > w:
            OPT[i, w] = OPT[i-1, w]
        else:
            OPT[i, w] = max{A[i] + OPT[i-1, w-A[i]], OPT[i-1, w]}

```

return OPT[n, W]

(b) No.
编码n需要的长度 $a = \Omega(n)$.
注意对n为item数 $\# \Rightarrow$

编码W需要的长度 $b = \Theta(\lg W)$

\Rightarrow time complexity. $O(nw) = O(a \cdot 2^b) \Rightarrow$ 增加时间

Problem 5:

8

$P = \{L \in \{0,1\}^*: \exists \text{ an alg. } A \text{ 可以在 polynomial-time 判定 } L\}$.

假设 alg. A_1 可以在 polynomial-time 判定 L_1 , alg. A_2 可以在 polynomial-time 判定 L_2 .

$\forall x \in L_1 \cup L_2 \Leftrightarrow x \in L_1 \text{ 或 } x \in L_2 \Rightarrow x \text{ 可以在多项式时间判定} \Rightarrow L_1 \cup L_2 \in P$ ✓

$\forall x \in L_1 \cap L_2 \Leftrightarrow x \in L_1 \text{ 且 } x \in L_2 \Rightarrow x \text{ 可以在多项式时间判定} \Rightarrow L_1 \cap L_2 \in P$. ✓

$L_1 L_2 = \{x_1 x_2 : x_1 \in L_1 \text{ 且 } x_2 \in L_2\}$

设 x 在 $L_1 L_2$ 中， x 需要 $C_1 n^k$ 的时间， A_1 判定不需要 $C_2 n^k$ 的时间 (n 是 x 的长度).

$x \in L_1 L_2$, 由上述 $L_1 L_2$ 的 definition, 首先花 $O(n)$ 时间将 x 分成 x_1 和 x_2 ($x_1 \in L_1$ 且 $x_2 \in L_2$) 的形式. ($n = |x|$)

x_1 和 x_2 可以在 $O(n^k)$ 的时间判定. \Rightarrow 判定 x 的 total time. ✓

$\forall x \in L_1 L_2$, 且 $|x|=n$, 考虑如下 alg. :

for i=0 to n:

$x_1 = x[1 \dots i]$

$x_2 = x[i+1 \dots n]$

If A_1 can decide x_1 in polynomial-time and A_2 can decide x_2 in polynomial-time:

return true

由上述 $L_1 L_2$ 的 definition, x 可以划分为 $x_1 x_2$ ($x_1 \in L_1$, $x_2 \in L_2$) 的形式 \Rightarrow 上述 alg. 会返回 true.

total-time: $O(n(n^k + n^k)) = O(n^{k+1}) \Rightarrow L_1 L_2 \in P$

* 模仿 Theorem 34.2 的证明。首先，由 TH 34.2， M_1 在 $O(n^k)$ 时间内 accept $L_1 \Rightarrow \exists \text{ constant } c$ s.t. ~~满足~~ 对 $\forall x \in L_1$, 执行 M_1 $c n^k$ 后, M_1 accept x .
 对 $\forall x' \in \overline{L}_1$, 有算法 M' , 执行 M' $c n^k$ 后, if M_1 accept x' , \checkmark then M' reject x' ; if M_1 has not accept x' , then M' accept x' by outputting 1. \Rightarrow 既然, $x' \notin L_1$, M' 会在 poly-time 内 accept $x' \Rightarrow \overline{L}_1 \in P$.

$$5. L_1^* = \{x \mid UL_1UL_1^2UL_1^3U\ldots\}$$

X

对 $\forall x \in L_1^*$, 不是由 L_1 中的 i 个符号(可重复)的拼接。 $L_1^{in} = \{uv \mid u \in L_1^i \wedge v \in L_1\}$.

$\Rightarrow \forall x \in L_1^*$

有如下 alg: $i=1$

while $i \leq n$:

for $j=i$ to n :

$$x' = x[i-j]$$

if M_1 can decide x' in polynomial time,

$$i=j+1$$

break

return true

由上述 Kleene closure 的定义, 上面的 alg. 最后必会返回 true. 设其 time complexity 为 $T(n)$.

则有 $T(n) = O(i^k) + T(n-i)$, where $1 \leq i \leq n$, 且 $T(0) = O(1) \Rightarrow T(n) \not\in \text{polynomial-time}$.

$\Rightarrow L_1^* \in P$.

Problem 6: $G_2(V_2, E_2)$

(a) $G_1(V_1, E_1)$ 和 $G_2(V_2, E_2)$ 是同构图 $\Leftrightarrow \exists \text{ 映射 } \sigma: V_1 \rightarrow V_2$ 使得 $\forall (u_i, u_j) \in E_1 \Leftrightarrow (\sigma(u_i), \sigma(u_j)) \in E_2$.

给定一个映射 $\sigma: |V_1| \rightarrow |V_2|$, 有如下 verify 过程.

也要耗时

Verify: for each $(u, v) \in E_1$,

if $(\sigma(u), \sigma(v)) \notin E_2$:
 output('no')

if no 'no' is printed during the above for-loop,
 output('yes')

\Rightarrow 上述 verify 算法在 $O(|E_1|E_2)$ 时间完成

\Rightarrow GRAPH-ISOMORPHISM $\in NP$

(b) To prove TAUTOLOGY $\in CONP \Leftrightarrow TAUTOLOGY\text{-complement} \in NP$.

\Leftrightarrow there exist a two-input poly time alg A and a constant c , s.t.

there exists a certificate y with $|y|=O(|x|^c)$ s.t. $A(x, y)=1$ for any $x \in TAUTOLOGY\text{-complement}$.

- for $x \notin \text{TAUTOLOGY}$, i.e. x ~~is TAUTOLOGY~~-complement, there is a certificate y of size polynomial in $|x|$ s.t. $A(x, y)$ correctly say 'No'.
- for $x \in \text{TAUTOLOGY}$, there is no certificate y for which $A(x, y)$ will say 'No'.
 定义 $A(x, y)$ say 'No' iff 在 x 的所有赋值中 $x=0$. Obviously, A 的运行时间为 $O(|x|^m)$, $\forall m$.
 polynomial time.

Problem 7: 10

(a) for any $L \in \text{co-NP} \Leftrightarrow \bar{L} \in \text{NP}$.

Since class P is closed under complement, then for any $L \in P$, we have $\bar{L} \in P$.

Notice $P \subseteq \text{NP}$, hence $\bar{L} \in \text{NP} \Rightarrow L \in \text{co-NP}$. Therefore, we have $P \subseteq \text{co-NP}$.

(b) 假设 $P = NP$, since P is closed under complement $\Rightarrow NP$ is closed under complement
 $\Leftrightarrow NP = \text{co-NP} \Rightarrow$ 矛盾
 故 $P \neq NP$.

Problem 6 (b):

To prove $\text{TAUTOLOGY} \in \text{coNP} \Leftrightarrow \text{TAUTOLOGY-complement} \in \text{NP}$.

Define $A(x, y) = 1$ iff the assignment y to input variables x_1, \dots, x_k makes x evaluate to 0.
 显然, alg. $A(\cdot, \cdot)$ 的时间复杂度是 polynomial 的。

$\forall x \in \text{TAUTOLOGY}$ -complement, 显然, 存在 \exists 一个 assignment y s.t. x evaluate to 0. (反证)

2) $x \in \text{TAUTOLOGY}$, i.e. there exists a certificate y with $|y| = k = O(|x|)$ s.t. $A(x, y) = 1$.

$\Rightarrow \text{TAUTOLOGY}$ -complement $\in \text{NP}$

$\Leftrightarrow \text{TAUTOLOGY} \in \text{coNP}$