算法基础

第十次作业（DDL：2024 年 12 月 19 日 23:59）

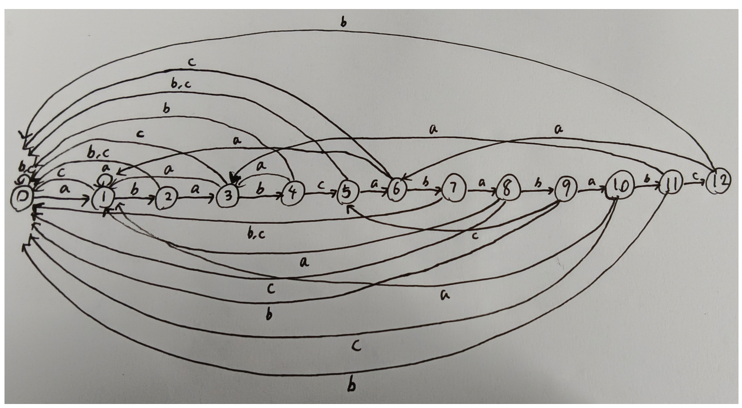
解答过程中请写出必要的计算和证明过程

Q1.(20 分) 对字母表 $\Sigma = \{a, b, c\}$，模式 P = ababcabababc，

1. 画出 P 对应的字符串匹配自动机的状态转换图。
2. 计算 P 的前缀函数 π 。

Solution:

1.



2.

P: a b a b c a b a b a b c

π: 0 0 1 2 0 1 2 3 4 3 4 5

Q2.(15 分) 定义一个字符串 $s_1, s_2, ..., s_n$ 的循环变换为 $s_2, s_3, ..., s_n, s_1$。给的两个字符串 P_1, P_2，给出 $O(n)$ 的算法确定 P_2 是否可以通过对 P_1 进行若干次循环变换得到。

Solution:

若 P_2 可以通过对 P_1 进行若干次循环变换得到，则 P_2 是 P_1P_1 的子串，反之亦然。以 P_2 为模式串，P_1P_1 为文本串运行一次 kmp 算法，输出匹配结果即可。时间复杂度 $O(n)$。

```
can_cycle(P1, P2)
```

```
    output kmp(P1P1, P2)
```

Q3.(15 分) 定义一个字符串 $S = s_1, s_2, ..., s_n$ 的最小循环节为最短的字符串 $T = s_1, s_2, ..., s_r$ ，且满足：

1. 是 S 的一个前缀。
2. 可以将 T 复制有限次得到字符串 $S' = TT...T$，使得 S 是 S' 的前缀。

例如，$ababa$ 的最小循环节是 ab，$abcabc$ 的最小循环节是 abc 。给定字符串 P ，给出 $O(n)$ 的算法确定 P 的最小循环节。

Solution:

假设 P 长度为 n，最小循环节长度为 t，则 P[1:(n-t)] = P[t+1:n]，所以 next[n] ≥ n - t。同理，n - next[n] 也一定是 P 的一个循环节，所以 n - next[n] ≥ t。所以 t = n - next[n]。对 P 运行一次 kmp 算法，n - next[n] 就是 P 的最小循环节。时间复杂度 $O(n)$。

```
min_cycle(P)
    next = kmp(P)
    n = length of P
    output n - next[n]
```

Q4.(20 分) 将 3-SAT 问题归约到独立集问题。

note. 3-SAT 问题：输入一个合取范式形式的命题公式 Φ，其中每个子句恰好包含 3 个文字，Φ 是否可满足？独立集问题：输入一个图 G 和一个整数 k，图 G 是否存在一个大小至少为 k 的独立集？

Solution: 令 Φ 为一个有 n 个变量和 m 个子句的 3-SAT 问题的实例。按照如下方式构造 G 和 k：

- 对于 3-SAT 公式 Φ 的每个子句，创建一个对应的三角形子图。这个子图中的三个顶点分别代表子句中的三个文字。
- 在同一个子句内部的三个顶点之间连边，使其成为一个三角形。
- 如果两个顶点对应的文字在不同的子句中但构成一个互补对（如一个是 x，另一个是 $\neg x$），在它们之间增加一条边。

- 设置 $k = m$，如果能在图中找到大小至少为 k 的独立集，则意味着每个子句至少能选取一个文字为真，即原来的 3-SAT 公式是可满足的。

eg.

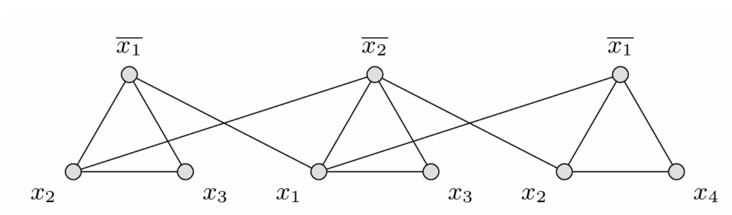


图 1: $\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$

Q5.(30 分) 考虑集合覆盖问题 (set cover problem)：

令 U 是一个包含 n 个元素的集合。令 $S = \{S_1, S_2, \ldots, S_m\}$ 是 U 的子集集合，且满足 $\bigcup_{i=1}^{m} S_i = U$。我们的目标是从 S 中选择尽可能少的子集，使得它们的并集覆盖 U。

考虑以下算法 SetCover 用于解决这个问题：

Algorithm 1: SetCover

Input: A set U and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ such that $\bigcup_{i=1}^{m} S_i = U$

Output: A collection $C \subseteq \mathcal{S}$ that covers U

```
Initialize C ← ∅;
while U contains elements not covered by C do
    Find the set S_i containing the largest number of uncovered elements;
    Add S_i to C;
return C;
```

为了分析上述算法，令 $k = \text{OPT}$，即最优解中所需的集合数。令 $E_0 = U$，且令 E_t 表示在第 t 步之后尚未被覆盖的元素集合。

(a) (15 分) 证明 $|E_{t+1}| \leq |E_t| - |E_t|/k$。

(b) (15 分) 证明算法 SetCover 是一个用于解决集合覆盖问题的 $(\ln n + 1)$-近似算法。

提示：证明算法 SetCover 在 $\text{OPT} \cdot (\ln n + 1)$ 步内结束。

Solution:

证明：

(a) $k = \mathrm{OPT}$ 是最优解中的集合数目，即最优解用不超过 k 个集合可以覆盖每一个 E_t。在第 $t+1$ 步，算法总是在 S 中选择覆盖 E_t 中最多未覆盖元素的集合。这个集合至少覆盖了 E_t 中 $|E_t|/k$ 个元素。因为如果覆盖元素的数量小于 $|E_t|/k$，则最优解无法用 k 个集合覆盖 E_t，这与 OPT 的定义矛盾。因此，满足：

$$|E_t| - |E_{t+1}| \geq |E_t|/k \implies |E_{t+1}| \leq |E_t| - |E_t|/k$$

(b)
- 运行时间：每步花费 $O(mn)$ 时间。接下来证明算法在 $\mathrm{OPT} \cdot \ln n$ 步内结束。因此，该算法的运行时间是多项式时间。
- 正确性：根据 (a)，以及 $|E_0| = |U| = n$，我们有：

  $$|E_t| \leq n\left(1 - \frac{1}{k}\right)^t \quad \text{对于任何} t \geq 1$$

  这个结论可以用数学归纳法证明 $(|E_1| \leq |E_0| - \frac{|E_0|}{k} = n(1 - \frac{1}{k}), |E_2| \leq |E_1| - \frac{|E_1|}{k} = n(1-\frac{1}{k})^2, \cdots)$。特别地，当 $t = \mathrm{OPT} \cdot \ln n = k \ln n$ 时：

  $$|E_t| \leq n\left(1 - \frac{1}{k}\right)^{k \ln n} < n \cdot \left(\frac{1}{e}\right)^{\ln n} = 1$$

  这意味着算法在 $\mathrm{OPT} \cdot \ln n$ 步内结束。增加一项以考虑贪心选择中的非最优情况，得到误差项为 1。因此，集合 C 的大小满足：

  $$|C| \leq (\ln n + 1) \cdot \mathrm{OPT}$$

因此，算法 SetCover 是集合覆盖问题的一个 $(\ln n + 1)$-近似算法。