Homework7

洪方舟
2016013259
Email: hongfz16@163.com

2018 年 4 月 18 日

Exercise 35.5-2

首先如果给定一系列集合想要验证这些集合能否覆盖所有的元素，那么肯定是多项式时间的，因此该问题是 NP 问题。下面由顶点覆盖问题归约来证明集合覆盖问题是 NP 完全的。

给定一个图 $G=(V,E)$，不妨假设图 G 中没有孤立点，那么构造集合 $X=E$，通过下面的规则给出集合 X 上待选择的所有集合 S_i：如果 $e_1,e_2\in E$，且 e_1,e_2 至少共享同一个顶点，那么 $e_1,e_2\in S_i$。那么顶点覆盖问题的一个解就可以通过如下规则转化为集合覆盖问题的解：每个被选入的顶点，与之相连的所有边构成的集合 S_i 就被选入，最后在图 G 中所有的边都被覆盖则在集合 X 中所有的元素都被覆盖，并且由顶点覆盖问题的最少选择的性质可知选出来的集合的数量也是最少的。

从另一个方向考虑，给定一个集合 X，以及若干子集 S_i，构造图 $G=(V,X)$，$\forall e_j\in S_i$，将 e_j 全都用一个点 v_i 连接。对于集合 X 的一个覆盖问题的解就可以通过如下规则转化为图 G 的顶点覆盖问题的解：如果 S_i 入选，则将通过 S_i 构造出来的顶点 v_i 选入，那么由于集合 X 中的元素与图 G 的边一一对应，则将集合中所有元素覆盖就意味着图中所有边被覆盖，又因为集合覆盖问题的最小选择的性质，则由之构造的顶点覆盖问题的解也是最优的。

综上所述，由于顶点覆盖问题是 NP 完全的，并且可以从顶点覆盖问题归约到集合覆盖问题，因此集合覆盖问题也是 NP 完全的。 □

Exercise 35.5-3

时间复杂度为 $O(\sum_{S\in\mathcal{F}}|S|)$ 的 $GREEDY-SET-COVER$ 算法如下

GREEDY-SET-COVER-IMPROVE($\mathcal{F}$)

```
1  let A be a new array with length of Max(S.size)
2  for S ∈ F
3        do add S to A[S.size]
4  let L be a new array with length of ∪_{S∈F} S
5  for S ∈ F
6        do for l ∈ S
7              do add S to L[l]
8  let Result be an empty array
9  let Covered be an empty array
10 let curr = A.size
11 while curr > 0
12       do while A[curr] == ∅ && curr ≠ 0
13             do curr=curr-1
14          if curr == 0
15             do BREAK
16          let S_0 be any element in A[curr]
17          add S_0 to Result
18          remove S_0 from A[curr]
19          for l ∈ (S_0 − Covered)
20             do for S ∈ L[l]
21                   do remove S from A[S.size]
22                      S.size = S.size − 1
23                      add S to A[S.size]
24                add l to T
25 RETURN Result
```

Problem 27-2

a.

P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C,A,B)

```
1  let n = A.rows
2  if n == 1
3       do c_11 = c_11 + a_11 b_11
4    else partition A, B, C into n/2 * n/2 submatrices A_11, A_12, A_21, A_22, B_11, B_12, B_21, B_22, C_11, C_12, C_21, C_22
5         spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_11, A_11, B_11)
6         spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_12, A_11, B_12)
7         spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_21, A_21, B_11)
8         P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_22, A_21, B_12)
9         sync
10        spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_11, A_12, B_21)
11        spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_12, A_12, B_22)
12        spawn P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_21, A_22, B_21)
13        P-MATRIX-MULTIPLY-RECURSIVE-SPACE(C_22, A_22, B_22)
14        sync
```

b.

工作量 $T_1(n)$，可由下递推式表达：$T_1(n) = 8T_1(n/2) + \Theta(1)$，根据主定理可知工作量 $T_1(n) = \Theta(n^3)$；
持续时间 $T_\infty(n)$，可由下递推式表达：$T_\infty(n) = 2T_\infty(n/2) + \Theta(1)$，根据主定理可知持续时间 $T_\infty(n) = \Theta(n)$

c.

该算法的并行度为 $\Theta(n^2)$，如果忽略符号 Θ 中的常数因子，$1000 * 1000$ 的矩阵上的并行度为 10^6，而使用书中描述的方法的并行度为 $10^9/lg^2(1000)$，后者大约是前者的 20 倍。