Homework 1, additional problems, solution set

1.a. We want to show that there are constants $0 < c < d$ and an integer n_{cd} such that for all $n \geq n_{cd}$, $c \cdot g(n) \leq f(n) \leq d \cdot g(n)$.

Using the handout on big Oh bounds, we know that for $n \geq 1$, $\log n \leq n$. Also, for $n \geq 1$, $\log n \geq 0$. Thus $2n^2 \leq 2n^2 + n \log n \leq 3n^2$. Setting $c = 2$, $d = 3$, and $n_{cd} = 1$, it follows that for all $n \geq n_{cd}$, $c \cdot g(n) \leq f(n) \leq d \cdot g(n)$.

b. The answer is "no". To see this, let $f(n) = 2n$ and $g(n) = n$. Then $2^{f(n)} = 2^{2n}$ and $2^{g(n)} = 2^n$. So if $2^{f(n)} = O(2^{g(n)})$ then $2^{2n} = O(2^n)$; i.e. $4^n = O(2^n)$; but this is not true.

For if it were true, there would be a constant $c > 0$ and an integer $\overline{n}$ such that for all $n \geq \overline{n}$, $4^n \leq c \cdot 2^n$; or equivalently, $2^n \leq c$; but this does not hold for $n > \log c$, and consequently there cannot be a pair $(c, \overline{n})$ of the desired form.

2.a. Suppose we first combine S_1 and S_2 and then combine the result with S_3. The total number of steps performed is $s_1 s_2 + (s_1 + s_2) s_3 = s_1 s_2 + s_1 s_3 + s_2 s_3$. If instead, we started by combining S_1 and S_3, the cost would be the same, and likewise if we started by combining S_2 and S_3.

b. Suppose we first combine S_1 and S_2. Let the resulting list be named T. By part (a), the remaining cost is the same regardless of the order in which T, S_3 and S_4 are combined. Thus, the overall cost is $s_1 s_2 + (s_1 + s_2) s_3 + (s_1 + s_2) s_4 + s_3 s_4 = \sum_{1 \leq i < j \leq 4} s_i s_j$. Clearly, the cost remains the same regardless of which pair is combined first.

c. Consider some set S_i. Over the course of the algorithm it forms part of larger and larger sets. The sets it is combined with in turn are disjoint and their union is $\cup_{j \neq i} S_i$. Thus the contribution to the total operation cost of the combinings in which S_i is involved is $s_i \cdot \sum_{j \neq i} s_j$. Summing over all i would count each operation twice, once for the combining of S_j with S_i and once for the combining of S_i with S_j. Thus the total number of operations being performed is $\frac{1}{2} \sum_{i \neq j} s_i s_j = \sum_{1 \leq i < j \leq k} s_i s_j$. Clearly, this is the same regardless of the order in which the combinings are performed.