1 Exercise 1

1.

D^0, P^0:

$$\begin{pmatrix} 0 & 80 & 40 & 10 \\ 80 & 0 & 10 & 20 \\ 40 & 10 & 0 & 90 \\ 10 & 20 & 90 & 0 \end{pmatrix}$$

$$\begin{pmatrix} \emptyset & 1 & 1 & 1 \\ 2 & \emptyset & 2 & 2 \\ 3 & 3 & \emptyset & 3 \\ 4 & 4 & 4 & \emptyset \end{pmatrix}$$

D^1, P^1:

$$\begin{pmatrix} 0 & 80 & 40 & 10 \\ 80 & 0 & 10 & 20 \\ 40 & 10 & 0 & \mathbf{50} \\ 10 & 20 & \mathbf{50} & 0 \end{pmatrix}$$

$$\begin{pmatrix} \emptyset & 1 & 1 & 1 \\ 2 & \emptyset & 2 & 2 \\ 3 & 3 & \emptyset & \mathbf{1} \\ 4 & 4 & \mathbf{1} & \emptyset \end{pmatrix}$$

The answers are

$$\langle 40, 10, 0, \mathbf{50} \rangle$$

$$\langle 3, 3, \emptyset, \mathbf{1} \rangle$$

2 5. $\Theta(k)$

3

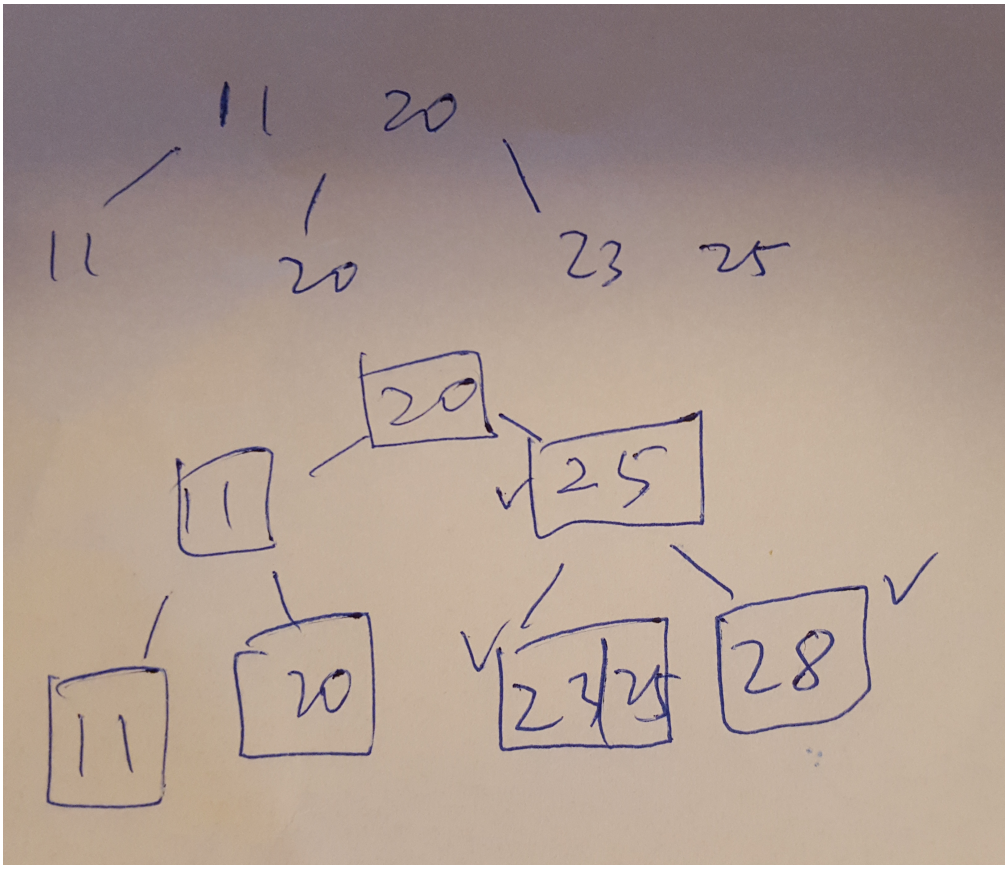
1. Vertical, left to right: GH vs. EF
2. Vertical, right to left: GH vs. EF

3. Horizontal, top to bottom: AB vs. CD

4. Horizontal, bottom to top: GH vs. EF

3.1 3
3.2 1 2 and 4

4 3



5
5.1 2
5.2 2
5.3 2, parallelism is 4
5.4 1,

2 Exercise 2

1

Sort A and B into monotonically decreasing order.

Here's a proof that this method yields an optimal solution. Consider any indices i and j such that $i < j$, and consider the terms ${a_i}^{b_i}$ and ${a_j}^{b_j}$. We want to show that it is no worse to include these terms in the payoff than to include ${a_i}^{b_j}$ and ${a_j}^{b_i}$, i.e., that ${a_i}^{b_i}{a_j}^{b_j} \geq {a_i}^{b_j}{a_j}^{b_i}$. Since A and B are sorted into monotonically decreasing order and $i < j$, we have $a_i \geq a_j$ and $b_i \geq b_j$. Since a_i and a_j are positive and $b_i - b_j$ is nonnegative, we have ${a_i}^{b_i - b_j} \geq {a_j}^{b_i - b_j}$. Multiplying both sides by ${a_i}^{b_j}{a_j}^{b_j}$ yields ${a_i}^{b_i}{a_j}^{b_j} \geq {a_i}^{b_j}{a_j}^{b_i}$.

Since the order of multiplication doesn't matter, sorting A and B into monotonically increasing order works as well.

2 Algorithm does not use backtracking: enumerate 2^n subsets, and compute the sum of each subset. In the worst case, a subset has $\Theta(n)$ elements, thus computing the sum of the subset is linear time. Then, $O(2^n \cdot n)$.

Algorithm uses backtracking: Assume that a_i is the i-th integer. configuration: (A, k, rT): A is the current subset, k indicates that we need to consider the k-th integer next time, rT is the remaining target.

For case 1, we consider the 1-st integer 2, and we generate two configurations ({2}, 2, 13) and ({}, 2, 15).

For case 2, we consider the 1-st integer 11, and we generate two configurations ({11}, 2, 4) and ({}, 2, 15).

When the remaining target is negative, it is a dead end. No need to continue further. When the remaining target is 0, it is a solution and we can return the corresponding A in the configuration as the result.

worst case, still $O(2^n \cdot n)$.

3 We have a source and a sink. We have N article vertices, one for each article. We have M reviewer vertices, one for each reviewer. In total, N+M+2 vertices.

Connect the source to each article vertex with an edge, where the capacity is 3.

Connect the sink to each reviewer vertex with an edge, where the capacity is 5.

Connect an article vertex with a reviewer vertex if the reviewer has the expertise to review the article.

If the maximum flow is not smaller than the $3N$, it is possible to assign each article with 3 reviewers.

For an edge that connect article vertex a_i and reviewer vertex r_j: 1 means that article a_i should be assigned to reviewer r_j.