British Informatics Olympiad Final

30 March – 2 April, 2000

Sponsored by Data Connection & SkillIT

Tournament

Romulus and Remus are keen players of games. Although they enjoy each others company they have decided to branch out and have joined a tournament. It is a standard knockout event. In the first round the competitors are paired and the winner of each game goes forward to the next round. This continues, from round to round, until there is only one player left. There are 2^k competitors, so at each round there will be an even number left.

Question 1.1
If there are 2^k competitors, how many games are played in total?

In this particular tournament everyone has a fixed ability; in any particular game the competitor with the higher ability will win. Furthermore, ability is transitive. In other words, if A is better than B and B is better than C it is also true that A is better than C.

Question 1.2
(a) The ability of the i^{th} competitor is `ability[i]`. Outline an algorithm, that is different to the knockout-competition algorithm, which determines the best competitor.
(b) How many games does your algorithm require for 2^k competitors?
(c) Does an algorithm exist which guarantees finding the best competitor, requiring fewer games than your new algorithm? Justify briefly.

Romulus is not too concerned about being the best, but he would like to know that he is not the worst. (Remus just wants to beat Romulus ...). The tournament, as it has currently been designed, does not provide this information.

Question 1.3
Suppose the tournament has already taken place and the results of each game are known. Assuming the ability of the competitors does not change, outline a further tournament for deciding who is the worst competitor. How many games does the additional tournament require?

The following year the organizers decide to make a tournament which determines both the best and worst competitors. They decide to start with two competitors and calculate the best and worst, then to add the remaining competitors one by one, recalculating the best and worst at each stage.

Question 1.4
(a) How are the best and worst , out of the first n (≥ 2) competitors, related to the best and worst out of the first $n + 1$ competitors.
(b) For 2^k competitors, what is the largest number of games this tournament can take? What is the smallest?

2. Second Best

The tournament is over and much to Romulus' pleasure he has come top. Remus is not so happy; he thought he was as good as Romulus, but he did much worse in the tournament. Over the next few days he plays games with all the other competitors and realizes that he is better than everyone except Romulus. With shock he realizes that it was Romulus who knocked him out in the first round; if he had been put against different opponents he might have won through to the final round.

Question 2.1

In a knockout tournament how many of the competitors may actually be better than the runner-up (the competitor who looses, in the last round, to the tournament winner.)

Remus can see one method of determining the second best in a tournament, by running another tournament with everyone except the best competitor. Unfortunately this requires to many additional games, and the tournament's committee are not willing to use it. Remus then discovers a *divide and conquer* algorithm which, for 2^k competitors, calculates the best competitor, along with a set of k competitors one of whom is the second best overall.

The algorithm works as follows. The competitors S are split into two groups L and R, each of size 2^{k-1}. For each of these groups the best competitor (`best(L)` and `best(R)`) is calculated, along with sets (each of size $k-1$) containing the second best in the group (`second(L)` and `second(R)`). These are then combined, to calculate the best competitor in S (`best(S)`) and a set (of size k) containing the second best in the group (`second(S)`). The values for groups L and R are generated using the same algorithm.

Question 2.2

Suppose `best(L)` $>$ `best(R)`. Write an expression for `max(S)` and `second(S)`.

Question 2.3

Outline code which uses this algorithm to run a tournament to calculates the best and second-best competitor.

The problem with these tournaments is that they fail to completely rank all the players. For example, if A beats B and C we still do not know whether B will beat C or vice versa. In a tournament with 16 competitors, or 1024 etc... if we calculate the best, second-best and worst, we only know the absolute position of 3 of the competitors.

Question 2.4

In which situation, if we calculate the best, second-best and worst, will we know the absolute position of more than 3 competitors?

Question 2.5

(a) Consider a tournament (of three games) where A beats B, C beats D and then A beats C. How many more games are necessary to completely rank all four competitors? Justify your answer.

(b) Suppose that, immediately after the three games of the tournament, E comes along. Show how to completely rank all five competitors with only four more games.