Simon Fraser University

Department of Computer Science

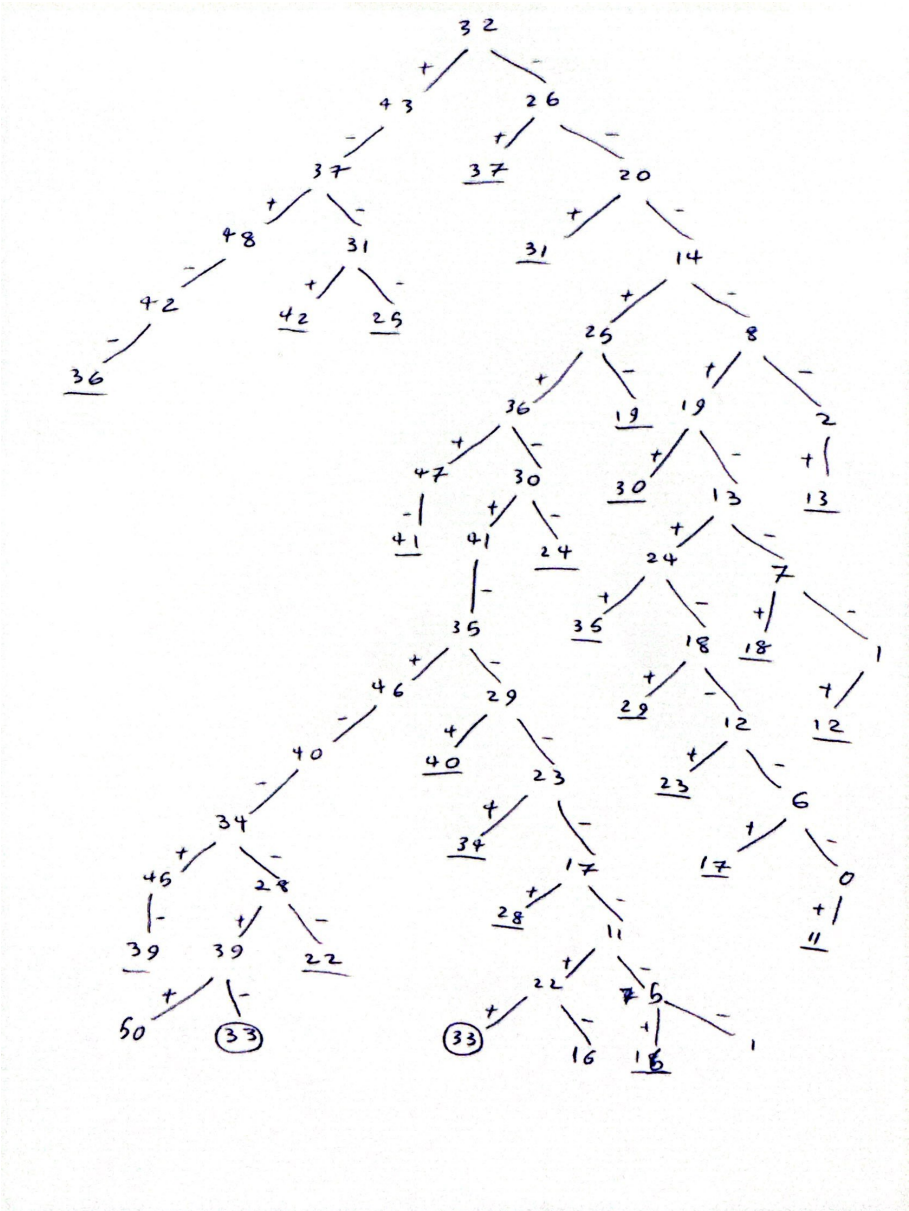
Bioinformatics Algorithms - Assignment 4

Name: Niloufar Saeidi

ID: 301590708

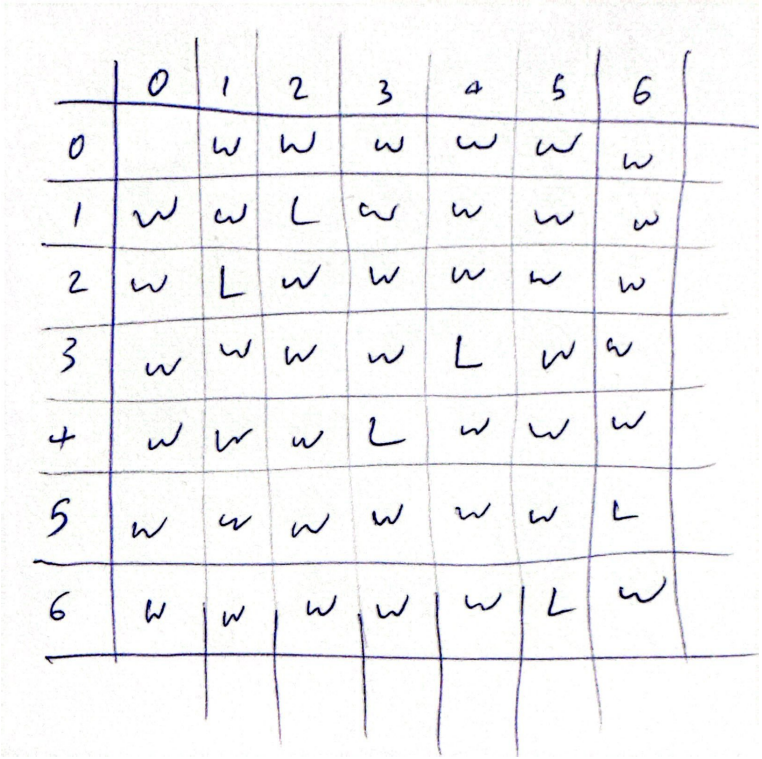
6.9

The dynamic programming approach for this problem is to use a graph in which each node j has at most two outgoing edges j+11 and j-6. If j+11>50, it does not exist, and if j-6<0 it does not exist. Then we consider a 50*50 table for saving the closest distance of each pair of two nodes. We can set all initial distances to a very large number and then start updating the values of this table from our starting node, which is in this case 32. Then update the distances by traversing our graph if we find a smaller distance. In the tree below, I have started from 32. The distances 32to43 and 32to26 are set to 1. If we go on and eliminate the repetitive numbers that we see, one of the shortest paths from level 32 to 33 will be 32, 26, 20, 14, 25, 36, 30, 41, 35, 46, 40, 34, 28, 39, 33. The length of all these shortest paths is 14. So 14 is the minimums number of buttons to be pressed.

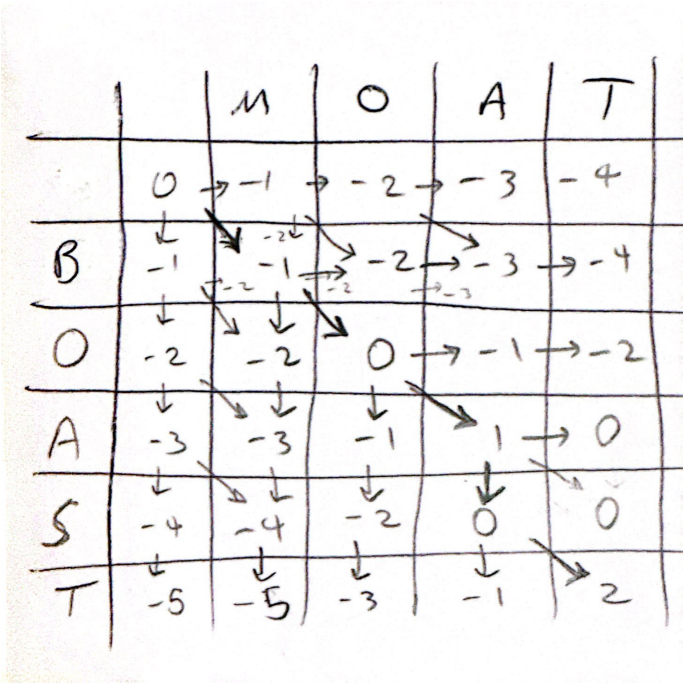


6.13

The row (0, x) for every x is win because player 1 can take all x nucleotides at once and win. Same with column (x, 0). Also, all the cells on the diagonals are win because the player can take all nucleotides at once. In the cells (2,1), every possible move of player 1 puts player 2 in one of the win cells. So (2,1) and (1, 2) are lose. In the cells (3,1) and (1,3), the player can take one nucleotide from the longer sequence and move the other player to the lose cells, so the cells are win. The same happens for all (x,1) and (1,x) cells with x>2 because they can put the other player in the positions (2,1) or (1,2) which are lose.Same happens for (2,x) and (x,2). In (3,4), (4,3), it is impossible to move the other player to the lose blocks and therefore these are lose. Then (3,x) and (x,3) for every x> 4 are win blocks. same for (4,x) and(x,4). Therefore, we end up at a pattern in which every block is win for player 1, except for $(n, m) = (2, 1), (1, 2), (3, 4), (4, 3), (6, 5), (5, 6), ..., (2x - 1, 2x), (2x, 2x - 1)$, where $x \geq 1$.



6.18

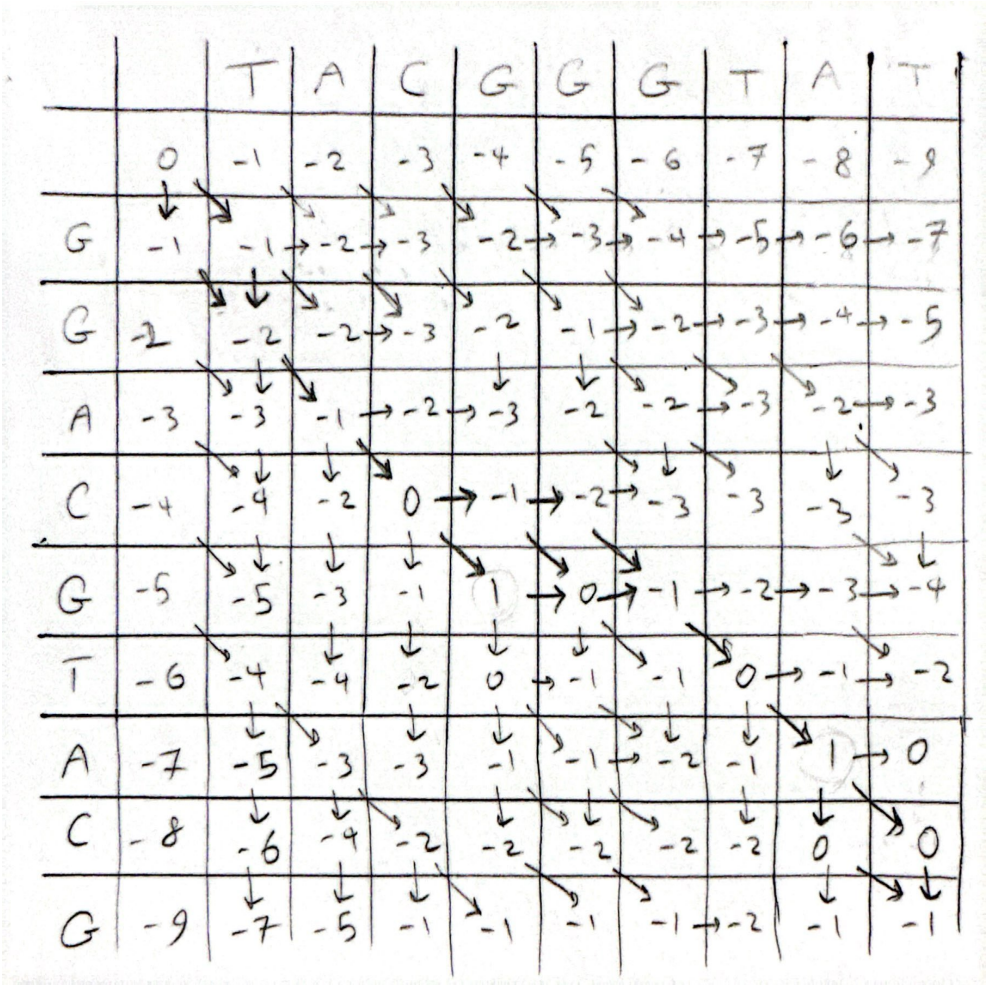


There is only one path from the starting point to (T, T) =2. This path represents the following alignment.

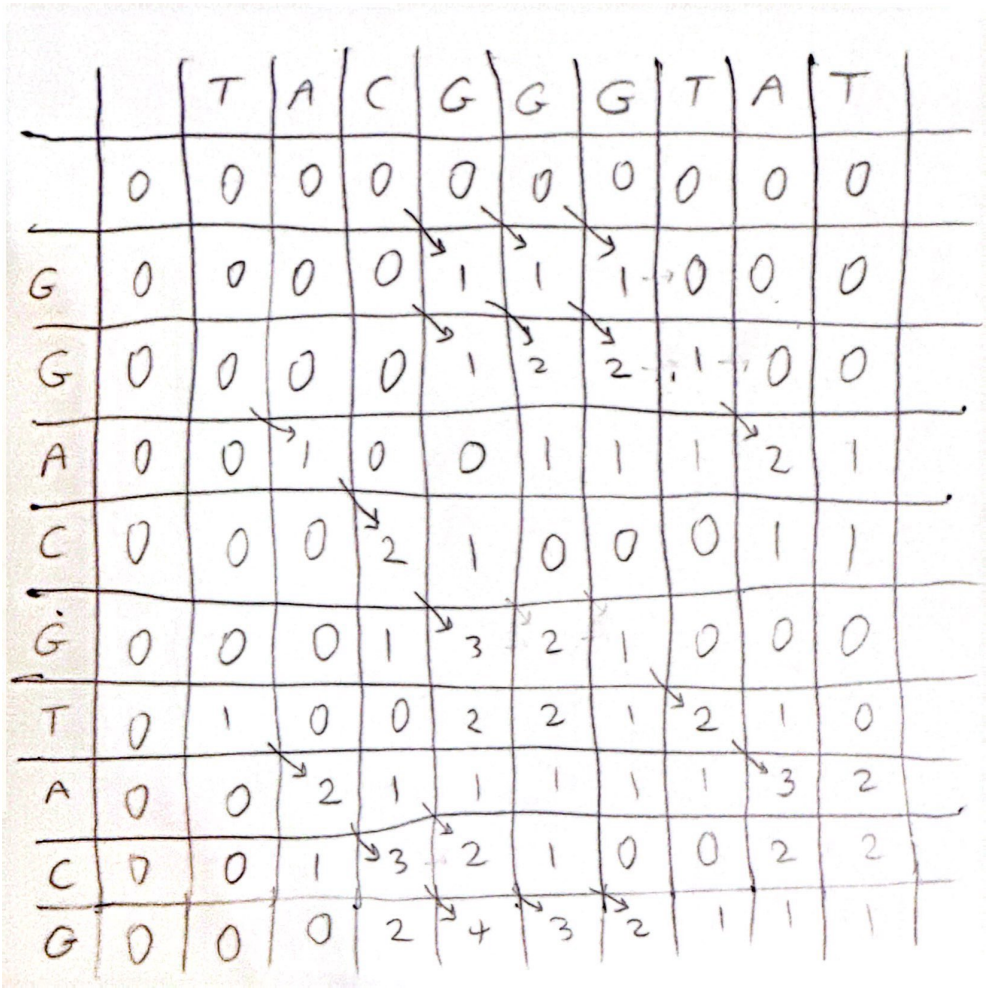
$$MOA - T$$

$$BOAST$$

6.20



There are 12 possible alignments(the bold ones). The score is -1.



		T	A	C	G	G	G	T	A	T
	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	1	1	1	0	0	0
G	0	0	0	0	1	2	2	1	0	0
A	0	0	1	0	0	1	1	1	2	1
C	0	0	0	2	1	0	0	0	1	1
G	0	0	0	1	3	2	1	0	0	0
T	0	1	0	0	2	2	1	2	1	0
A	0	0	2	1	1	1	1	1	3	2
C	0	0	1	3	2	1	0	0	2	2
G	0	0	0	2	4	3	2	1	1	1

The highest score local alignment(longest) is TACG with score 4.

If we suppose we use an affine gap penalty where it costs -20 to open a gap, and -1 to extend it and scores of matches and mismatches are unchanged, the optimal global alignment contains no indels and only matches and mismatches which is:

$$TACGGGTAT$$

$$GGACGTACG$$

This alignment gives us an score of $(-1) + (-1) + (-1) + (-1) + (+1) + (-1) + (-1) + (-1) + (-1) = -7$ corresponding to 4 mismatches, 1 match, and again 4 mismatches. This total score is higher than any other score belonging to the paths which open at least one gap, which applies the penalty of -20. Therefore, we can simply get to this conclusion of the optimal path without filling out the table values.

1

Proof that the Asymmetric Edge Recombination Operator preserves directed edges and common sub-strings between parent chromosomes.

When we are calculating the edge table, we have at most 2 and at least 1 entry for each element. If there are two entries in the table for an element, it means that the element had its outgoing edge to different neighbors in the each of the two parent sequences; which means that there was no common subsequence starting from that element in the parents. And the common subsequences, if exist, are in the elements with one entry in the edge table. The Asymmetric Edge Recombination Operator gives priority to the element with the fewest entries when choosing. This means that if there are any elements with one entry(same as if there are any common subsequences), they are chosen first and then their neighbor(which is the only option) is chosen. Therefore, all the common subsequences will be preserved.

2

Extend the regular Edge Recombination Operator to preserve common sub-strings between parents.

When the edge table is constructed, if an item is already in the edge table and we are trying to insert it again, that element of the sequence must be a common edge. The elements of a sequence are stored in the edge table as integers, so if an element is already present, the value is inverted: if A is already in the table, change the integer to -A. The sign acts as a flag. Consider the following sequences and edge table: a b c d e f and c d e b f a.

a: b, -f, c

b: a, c, e, f

c: b, -d, a

d: -c, -e

e: -d, f, b

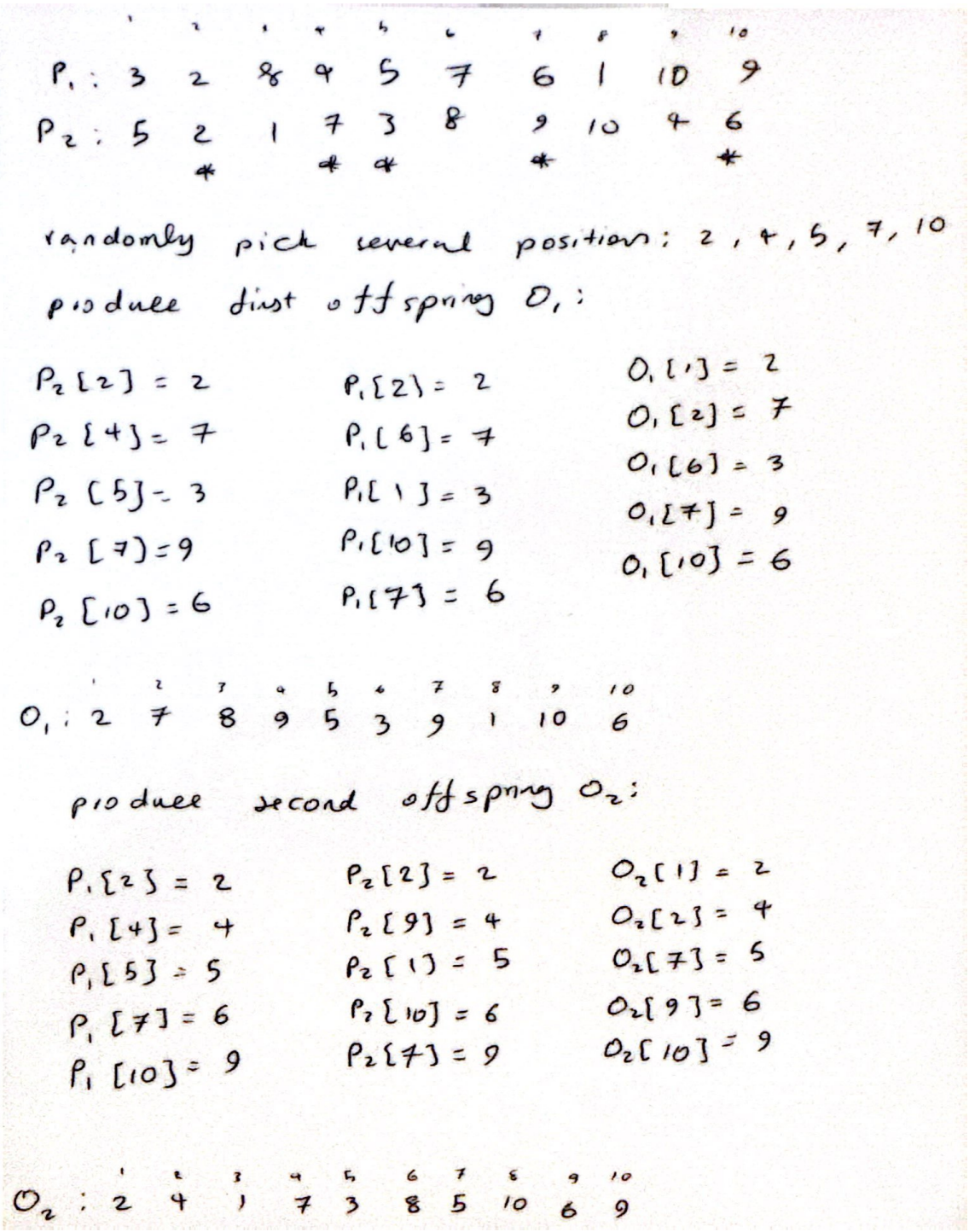
f: e, -a, b

The new edge table is the same as the old edge table, except for the flagged elements. One of three cases holds for an edge table entry. 1) If four elements are entered in the table as connections to a given table entry, that entry is not part of a common subsequence. 2) If three elements are entered as connections to a given table entry, then one of the first two elements will be negative and represents the beginning of a common subtour. 3) If only two elements are entered for a given table entry, both must be negative and that entry is an internal element in a common subsequence. Giving priority to negative entries when constructing offspring affects edge recombination for case 2 only. In case 1, no connecting

elements have negative values, and in case 3 both connecting elements are negative, so edge recombination behaves just as before. In case 2, the negative element which represents the start of a common subtour is given first priority for being chosen. Once this common subsequence is started, each internal element (case 3) of the sequence has only one edge in and one edge out, so it is guaranteed that the common sections of the sequence will be preserved.

3

Apply Order Crossover (OX2) to the following two permutations and produce two offspring. Show all your work. 3 2 8 4 5 7 6 1 10 9, 5 2 1 7 3 8 9 10 4 6



4

What is the survival probability of a schema H under crossover? Assume a string length of l and that the crossover point is chosen uniformly from among all possible choices. Give a general equation.

The survival probability of a crossover is the probability that the separating point does

not fall between the specified bits. The probability that the crossover point is between these is the longest number of distances between two specified points of the schema(length or $\delta(H)$) divided by the total number of distances in the schema(the number of bits - 1 $(l-1)$). So the survival probability P_s is:

$$P_s = 1 - \frac{\delta(H)}{l-1}$$

5

Then assume l = 10 and compute the exact survival probability for a schema of defining length 3, 5, and 8 [1 Mark each].

$$l = 3 : P_s = 1 - \frac{3}{9} = \frac{6}{9}$$

$$l = 5 : P_s = 1 - \frac{5}{9} = \frac{4}{9}$$

$$l = 8 : P_s = 1 - \frac{8}{9} = \frac{1}{9}$$