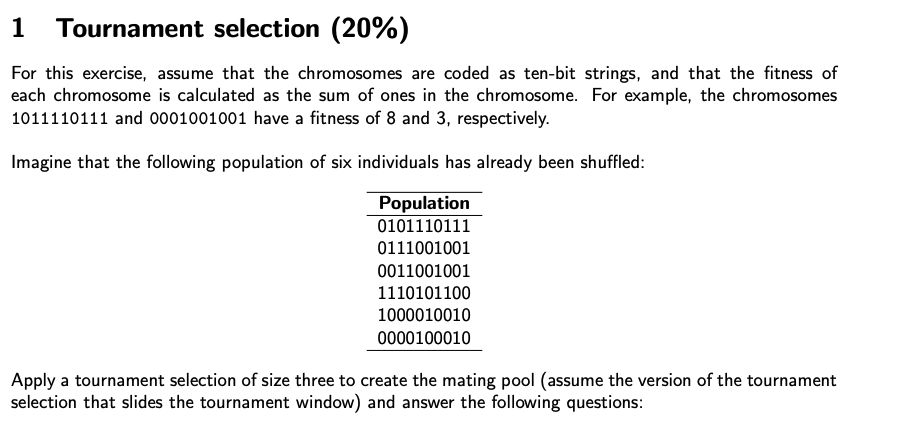
HW1\_EC

Jesus Eider Diaz Moraila

A00828174



tournament = c ("0101110111","0111001001","0011001001","1110101100","1000010010","0000100010")  
set.seed(42)  
  
random\_tournament = sample(tournament,6)  
f = vector()  
  
for (i in 1:length(random\_tournament) ) {  
 f[i]= table(strsplit(random\_tournament[i],"")) [2]   
}  
  
mating\_pool= vector()  
  
#slinding window of 3 elements  
  
for (i in 1:length(f) ) {  
 sliding\_window = seq(from=i,to=i+2,by=1)  
 overflow= i+2 - length(f)  
 if (overflow > 0) {  
 sliding\_window = c(sliding\_window[c(1:(3 - overflow))],1:overflow)  
 }  
 mating\_pool[i] = max(f[sliding\_window] )  
}

# • How many copies of each chromosome are present in the mating pool?

table(mating\_pool)

## mating\_pool  
## 6 7   
## 3 3

random\_tournament

## "0101110111" "1000010010" "0000100010" "1110101100" "0111001001" "0011001001"

#3 of the 1st and 3 of the 4th

# • What is the average fitness of the chromosomes in the mating pool?

mean(mating\_pool)

## [1] 6.5

# • If the tournament size is reduced to one, what is the probability that the chromosome 1110101100 appears in the mating pool?

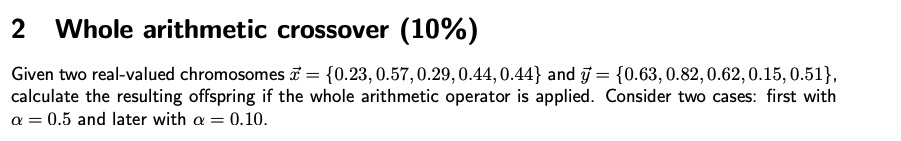
#if the ournament size is reduced to one means there is no comptetition so it will be 1/n and in this case 1/6 probability to appear in the mating pool

# • If the tournament size is increased to five, what is the probability that the chromosome 0111001001 appears in the mating pool?

tournament = c ("0101110111","0111001001","0011001001","1110101100","1000010010","0000100010")  
set.seed(42)  
  
random\_tournament = sample(tournament,6)  
f = vector()  
  
for (i in 1:length(random\_tournament) ) {  
 f[i]= table(strsplit(random\_tournament[i],"")) [2]   
}  
  
mating\_pool= vector()  
  
#slinding window of 3 elements  
  
for (i in 1:length(f) ) {  
 sliding\_window = seq(from=i,to=i+4,by=1)  
 overflow= i+4 - length(f)  
 if (overflow > 0) {  
 sliding\_window = c(sliding\_window[c(1:(5 - overflow))],1:overflow)  
 }  
 mating\_pool[i] = max(f[sliding\_window] )  
}  
  
mating\_pool

## [1] 7 6 7 7 7 7

# 0 since it must the top 2 in oder to appear



x = c(0.23, 0.57, 0.29, 0.44, 0.44)  
  
y = c(0.63, 0.82, 0.62, 0.15, 0.51)

#ui= alpha\* xi + (1-alpha)\*yi  
alpha= 0.5  
u\_0.5 = (alpha \* x) + (1-alpha)\*y  
v\_0.5= (alpha\* y) + (1-alpha)\*x  
#results for alpha = 0.5  
u\_0.5

## [1] 0.430 0.695 0.455 0.295 0.475

v\_0.5

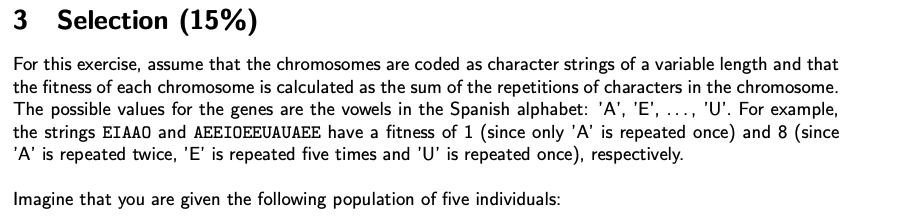
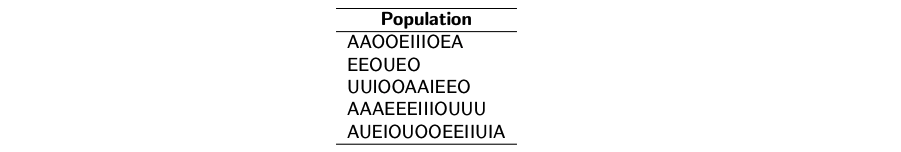
## [1] 0.430 0.695 0.455 0.295 0.475

alpha= 0.1  
u\_0.1 = (alpha \* x) + (1-alpha)\*y  
v\_0.1= (alpha\* y) + (1-alpha)\*x  
#results for alpha = 0.1  
u\_0.1

## [1] 0.590 0.795 0.587 0.179 0.503

v\_0.1

## [1] 0.270 0.595 0.323 0.411 0.447

Population = c("AAOOEIIIOEA", "EEOUEO" ,"UUIOOAAIEEO","AAAEEEIIIOUUU","AUEIOUOOEEIIUIA")  
fitness = vector()   
for (i in 1:length(Population)){  
 fitness[i] = sum(floor(table(strsplit(Population[i],""))/2))  
  
}

## • Calculate the probabilities of selecting each one of these individuals, based on a proportional selection (based on the fitness).

prop\_selection= print(fitness / sum(fitness))

## [1] 0.1904762 0.0952381 0.2380952 0.1904762 0.2857143

## • Calculate the probabilities of selecting each one of these individuals, based on a linear ranking selection with C = 2.

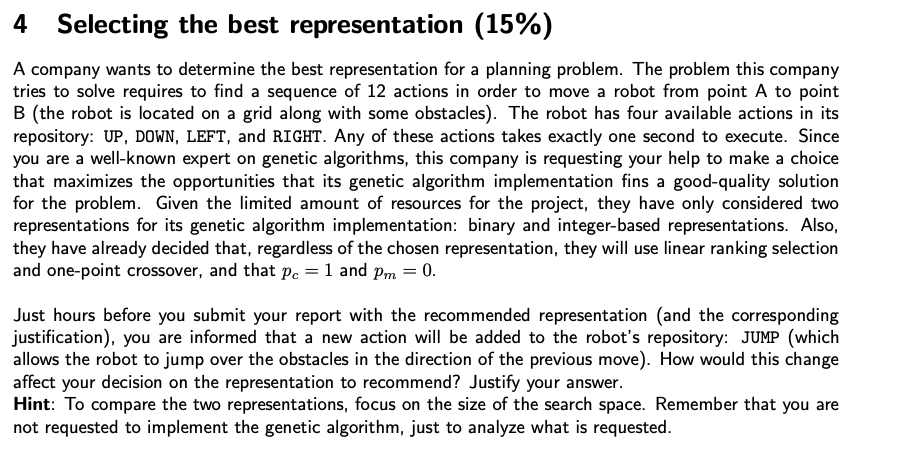
r = vector()  
c = 2  
n\_1 = (length(Population)-1)  
  
for (i in 1:length(fitness) ){  
   
 r[i] = which(sort(fitness) == fitness[i])[1] -1   
  
}  
  
linear\_ranking\_fitness = print(r \* (c/n\_1))

## [1] 0.5 0.0 1.5 0.5 2.0

## • Calculate the probabilities of selecting each one of these individuals, based on an exponential ranking selection with m = 3.

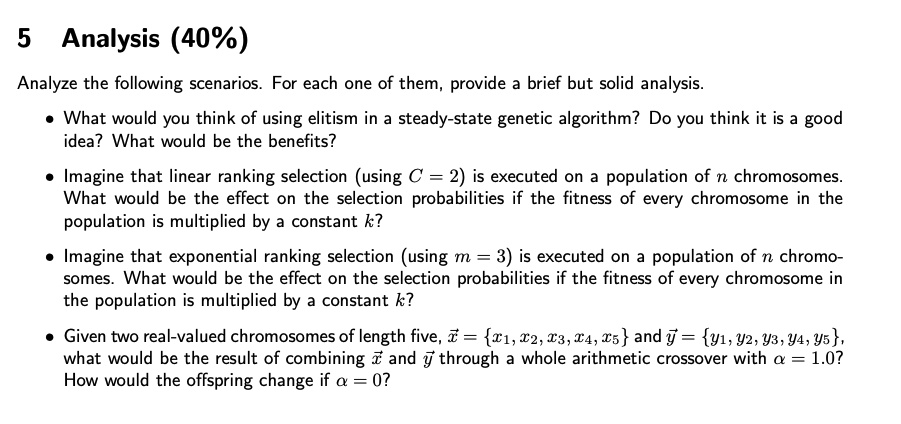
m = 3  
  
exponential\_ranking\_fitness = print(m \* (r/n\_1) ^ (m-1))

## [1] 0.1875 0.0000 1.6875 0.1875 3.0000



In order to represent in binary we need a solution that is formed of the following tuple (0,1,0,0,0) where each tuple means a command in this case this encoding means a DOWN action (1,0,0,0,0) means a UP action (0,0,0,0,1) means a JUMP action an so on, the final solution representation will be (1,0,0,0,0) twelve times meaning we have a 2 ^(5\*12) possible solutions or a search space of 115292150 4606846976 maybe less if we don’t accept a solution that have two ones in a window of five, like (1,0,0,0,1) this is not a possible piece of a solution.

On the other hand we have a integer representation that is simply look like (1,2,1,2,4,5,3,3,1,1,2,1) where 1 means UP, 2 means DOWN an so on, in this case we have a 5^12 solutions or a search space of 244140625 .



1. I f we are concern about rapid convergence the combination of steady-state and elitism will be a good choice, since every cycle we substitute the worst individuals and then passing the best individual every time, this will cause every cycle the population will be more likely to be akin the “elite solution” , on the other hand this rapid convergence may cause that the algorithm gets stuck in a local optima because this combination is not in favor of diversity.
2. Nothing since the fitness is calculated by the relative ranking of the population and if we multiply the whole fitness by any constant the fitness ranking is preserved
3. Same as 2) we no compute the selection based on the direct fitness but in the ranking, and because the ranking is conserved if the fitness is multiplied by a constant, everything remains the same
4. The result offspring of both alpha values will be the parents in both cases, since the crossover formula dictates the proportion of each parent in the offspring, thus using a alpha=1 will give us 100% proportion of the first parent + 0% of the second parent for the first offspring and 0% of the first parent and 100% for the second parent for the second offspring. And for the case of alpha = 0 it’s the same but the order is inverse in the offspring.