MATH4406 – HW3

Madeleine Nargar, 41214538

Question 2a

All ten problems solved resulted in a (σ, Σ) policy as optimal i.e. $d_t(s) = \begin{cases} \Sigma - s & s \leq \sigma \\ 0 & s > \sigma \end{cases}$.
Results of the backward induction algorithm can be seen in the appendix.

Optimal (σ, Σ) policies found (for problems with $O(u) = cu + \mathbf{1}\{u > 0\}K$)

M	c	K	$h(u)$	$g(u)$	$f(u)$	p_j	σ	Σ
2	1	2	u	0	10u	(0.5,0.25,0.25)	$\begin{cases} 1 & t < 13 \\ 0 & t = 13, 14 \end{cases}$	3
5	2	2	u	0	10u	(0.25, 0.25,0.25,0.25,0,0)	$\begin{cases} 1 & t < 14 \\ 0 & t = 14 \end{cases}$	$\begin{cases} 3 & t < 14 \\ 2 & t = 14 \end{cases}$
5	1	5	u	0	10u	(0.25, 0.25,0.25,0.25,0,0)	$\begin{cases} 1 & t < 14 \\ 0 & t = 14 \end{cases}$	$\begin{cases} 5 & t < 12 \\ 4 & t = 12 \\ 3 & t = 13, 14 \end{cases}$
5	1	2	4u	0	15u	(0.25, 0.25,0.25,0.25,0,0)	2	1
5	1	2	u	0	15u	(0.1, 0.1,0.3,0.3,0.1,0.1)	$\begin{cases} 3 & t < 14 \\ 2 & t = 14 \end{cases}$	$\begin{cases} 5 & t < 14 \\ 4 & t = 14 \end{cases}$
5	1	2	u	3u	6u	(0.1, 0.1,0.3,0.3,0.1,0.1)	$\begin{cases} 1 & t < 12, t = 13 \\ 2 & t = 12 \\ 3 & t = 14 \end{cases}$	$\begin{cases} 5 & t \neq 13 \\ 4 & t = 13 \end{cases}$
5	1	2	0	3u	6u	(0.1, 0.1,0.3,0.3,0.1,0.1)	$\begin{cases} 2 & t < 14 \\ 4 & t = 14 \end{cases}$	5
5	1	2	0	0	6u	(0.1, 0.1,0.3,0.3,0.1,0.1)	$\begin{cases} 2 & t \neq 13 \\ 3 & t = 13 \end{cases}$	$\begin{cases} 5 & t < 14 \\ 4 & t = 14 \end{cases}$
5	2	2	2u	5u	30u	(0.7, 0.2,0.1,0,0,0)	$\begin{cases} 1 & t < 11 \\ 0 & t = 11, 12, 13 \\ 2 & t = 14 \end{cases}$	$\begin{cases} 2 & t < 14 \\ 5 & t = 14 \end{cases}$
5	2	0	0.5u	0	6u	(0.1, 0.1,0.3,0.3,0.1,0.1)	$\begin{cases} 3 & t < 14 \\ 2 & t = 14 \end{cases}$	$\begin{cases} 4 & t < 14 \\ 3 & t = 14 \end{cases}$

Question 2b

For the problem with $N = 15, M = 5,\ h(u) = u,\ g(u) = 0,\ f(u) = 6u,\ p_j = (0.1, 0.1, 0.3, 0.3, 0.1, 0.1)$, a non-$(\sigma, \Sigma)$ threshold policy results from using the ordering cost $O(u) = \begin{cases} u & u < 3 \\ 2u - 5 & u \geq 3 \end{cases}$.

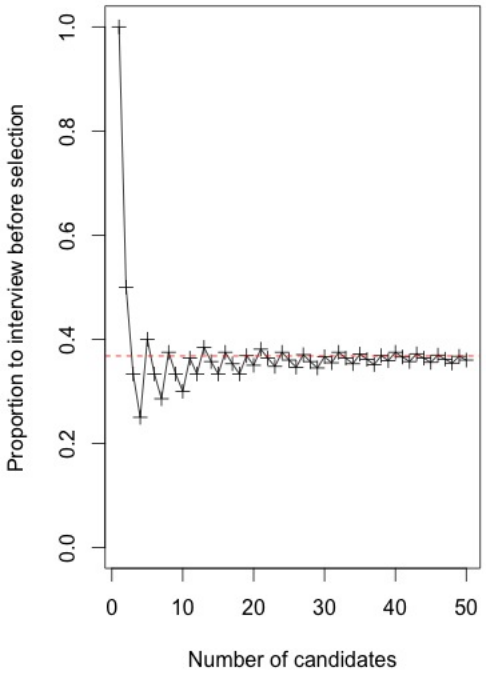
(see code in `inventory.R` - optimal policy and value produced using backwards induction)

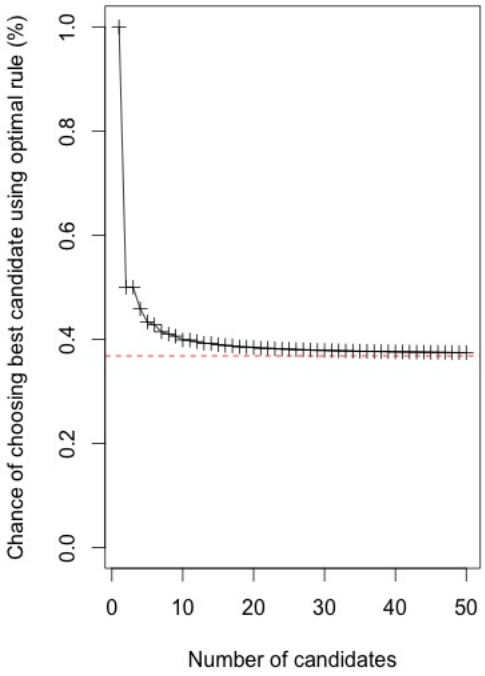
```
> #PART b: different order cost
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1),
+              o.=function(u){ifelse(u>=3, 2*u - 5, u)} ,
+              g.=function(u) {0}, h.=function(u) {u}, f.=function(u) {10*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
1 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
2 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
3 1 1 1 1 1 1 1 1 1  1  1  1  1  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
```

With this ordering cost (perhaps a 'with kickbacks' ordering cost), the optimal policy is to order three units when possible, meaning there is no target inventory level Σ.

Question 4a)

See code in `secretary.R` - optimal policy and value produced using backwards induction.





Question 4b)

Starting from p.101, It is optimal to continue while:

$$u_t^*(0) = \frac{t}{N}[\frac{1}{t} + \frac{1}{t+1} \cdots \frac{1}{N-1}] > \frac{t}{N}$$

$$\Rightarrow \tau = \max_{t \geq 1}\{[\frac{1}{t} + \frac{1}{t+1} \cdots + \frac{1}{N-1}] > 1\} \tag{4.6.11}$$

For N sufficiently large, assume there exists $\tau(N)$ such that:

$$\frac{1}{\tau(N)} + \frac{1}{\tau(N)+1} \cdots + \frac{1}{N-1} \approx 1 \tag{1}$$

$$\Rightarrow u^*_{\tau(N)}(0) \approx \frac{\tau(N)}{N} \tag{2}$$

$$\frac{1}{\tau(N)} + \frac{1}{\tau(N)+1} \cdots + \frac{1}{N-1} = \sum_{t=\tau(N)}^{N-1} \frac{1}{t}$$

$$\approx \int_{\tau(N)}^{N} \frac{1}{x} dx \qquad \text{for large N}$$

$$= \log(\frac{N}{\tau(N)}) \qquad (3)$$

$$\log(\frac{N}{\tau(N)}) \approx 1 \Rightarrow \lim_{N\to\infty} \frac{N}{\tau(N)} = e$$

$$\Rightarrow \frac{\tau(N)}{N} \to \frac{1}{e}$$

By (4.6.10) and (2), $u_1^*(0) = u_1^*(0) = u_{\tau(N)}^*(0) \approx \frac{\tau(N)}{N} \to \frac{1}{e}$.
i.e. Yielding the rule that for large N, N/e candidates should be interviewed, then after that the next candidate that is the best so far should be chosen, which will result in the best candidate being hired with a probability of approximately $1/e$.

You could use this rule in any situation with a similar structure, for example, buying a car, finding a house in which the assumptions (no possibility to recall past candidates, no search cost, objective to maximise chance to choosing the absolute best) are reasonable. In my recent life behaved somewhat like this when buying a car, but stopped significantly earlier than recommended when looking for a house - perhaps because search costs were non-negliable, making satisficing reasonable alternative.

Question 4c)

Chow[1] considered a variation of the Secretary problem in which the object is to minimise the expected value of the absolute rank on the applicant selected (rather than maximising the probability of choosing the highest ranked applicant, as in the classic secretary problem). Chow finds that the optimal stopping policy is to stop at the first $i \geq 1$ for which the relative rank of the i^{th} applicant (compared to already seen applicants) is less than or equal to s_i, where $(s_1, s_2, \ldots, s_n)$ can be found by:

$$s_i = \begin{cases} n & i = n \\ c_i \frac{i+1}{n+1} & i = n-1, \ldots, 1 \end{cases}, \qquad c_{i-1} = \begin{cases} \frac{n+1}{2} & i = n \\ \frac{1}{i}\left(\frac{n+1}{i+1}\frac{s_i(s_i+1)}{2} + (i - s_i)c_i\right) & i = n-1, \ldots, 1 \end{cases}$$

Here c_0 is the expected (absolute) rank of the applicant selected, which tends to approximately 3.9 as $n \to \infty$.

[1] Y. Chow, Optimal selection based on relative rank (the 'secretary problem'): *Israel journal of mathematics*, **2**, 81-90 (1964).

Appendix

Optimal policies for Q2a:

```
> #1
> inventoryMDP(N=15, M=2, p = c(.5,.25,.25),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u) {10*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 2 2 2 2 2 2 2 2  2  2  2  2  2 NA
1 1 1 1 1 1 1 1 1 1  1  1  1  0  0 NA
2 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #2
> inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
+              o.=function(u){ifelse(u==0,0,2*u+2)} ,
+              g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u) {10*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 3 3 3 3 3 3 3 3 3  3  3  3  3  2 NA
1 2 2 2 2 2 2 2 2 2  2  2  2  2  0 NA
2 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #3
> inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
+              o.=function(u){ifelse(u==0,0,1*u+5)} ,
+              g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u) {10*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 5 5 5 5 5 5 5 5 5  5  5  4  3  3 NA
1 4 4 4 4 4 4 4 4 4  4  4  3  2  0 NA
2 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #4
> inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {0}, h.=function(u) {4*u}, f.=function(u) {15*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 2 2 2 2 2 2 2 2  2  2  2  2  2 NA
1 1 1 1 1 1 1 1 1 1  1  1  1  1  1 NA
2 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
```

```
> #5
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u) {15*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 5 5 5 5 5 5 5 5 5  5  5  5  5  4 NA
1 4 4 4 4 4 4 4 4 4  4  4  4  4  3 NA
2 3 3 3 3 3 3 3 3 3  3  3  3  3  2 NA
3 2 2 2 2 2 2 2 2 2  2  2  2  2  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #6
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {3*u}, h.=function(u) {1*u}, f.=function(u) {6*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 5 5 5 5 5 5 5 5 5  5  5  4  5  4 NA
1 4 4 4 4 4 4 4 4 4  4  4  3  4  3 NA
2 0 0 0 0 0 0 0 0 0  0  3  0  3  2 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  2  1 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #7
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {3*u}, h.=function(u) {0*u}, f.=function(u) {6*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 5 5 5 5 5 5 5 5 5  5  5  5  5  5 NA
1 4 4 4 4 4 4 4 4 4  4  4  4  4  4 NA
2 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  0  2 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  1 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #8
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
+              o.=function(u){ifelse(u==0,0,1*u+2)} ,
+              g.=function(u) {0*u}, h.=function(u) {0*u}, f.=function(u) {6*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 5 5 5 5 5 5 5 5 5  5  5  5  5  4 NA
1 4 4 4 4 4 4 4 4 4  4  4  4  4  3 NA
2 3 3 3 3 3 3 3 3 3  3  3  3  3  2 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  2  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
```

```
>
> #9
> inventoryMDP(N=15, M=5, p = c(0.7,0.2,0.1,0,0,0 ),
+              o.=function(u){ifelse(u==0,0,2*u+2)} ,
+              g.=function(u) {5*u}, h.=function(u) {2*u}, f.=function(u) {30*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 2 2 2 2 2 2 2 2  2  2  2  5 NA
1 1 1 1 1 1 1 1 1 1  1  0  0  4 NA
2 0 0 0 0 0 0 0 0 0  0  0  0  3 NA
3 0 0 0 0 0 0 0 0 0  0  0  0  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0 NA
>
>
> #10
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1),
+              o.=function(u){ifelse(u==0,0,2*u)} ,
+              g.=function(u) {0}, h.=function(u) {0.5*u}, f.=function(u) {6*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 4 4 4 4 4 4 4 4 4  4  4  4  4  3 NA
1 3 3 3 3 3 3 3 3 3  3  3  3  3  2 NA
2 2 2 2 2 2 2 2 2 2  2  2  2  2  1 NA
3 1 1 1 1 1 1 1 1 1  1  1  1  1  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
> #PART b: different order cost
> inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1),
+              o.=function(u){ifelse(u>=3, 2*u - 5, u)} ,
+              g.=function(u) {0}, h.=function(u) {u}, f.=function(u) {10*u})
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
1 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
2 3 3 3 3 3 3 3 3 3  3  3  3  3  3 NA
3 1 1 1 1 1 1 1 1 1  1  1  1  1  0 NA
4 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
5 0 0 0 0 0 0 0 0 0  0  0  0  0  0 NA
>
```

```
###inventory.R - hw3: q2 inventory contol####
cat("\014")
rm(list =ls ())

#backward induction function
inventoryMDP <- function(N,M,o.,f.,h.,g.,p){
  #set up
  m <- M+1
  S <- 1:m #states
  A <- 1:m  #actions
  #transition matrix
  P <- array(data=0,dim=c(m,m,m))  #transition matrix
  for (s in S) {
    for (a in A){
      if ((s+a) <= (M+2)) {
        for (j in S) {
          if (j <= (s+a-1)){
            P[s,j,a] <- p[s+a-j]   #fill from p
          }
        }
        P[s,1,a] <- 1 - sum(P[s,-1,a])  #demand > s+a -> j=0
      }
    }
  }

  #rewards
  F. <- function(u){  #revenue function
    ifelse(u<=M,sum(f.(0:max(0,(u-1)))*p[1:u]) + f.
(u)*sum(p[(u+1):m]),0)
  }
  R <- array(0,dim=c(m,m))   #reward array
  for (s in S) {
    for (a in A) {R[s,a] <- -h.(s+a-2) - o.(a-1) + F.(s+a-2)}
  }

  ###run policy eval alg
  U <-
array(1,dim=c(m,N,2),dimnames=list(c(0:M),c(1:N),c("policy","value")))
  u <- array(1,dim=c(m,m))

  t <- N
  for (s in S) {
  U[s,t,1] <- g.(s-1)
  }
  U[,t,2] <- NA

  for (t in (N-1):1){
    for (s in S){
      for (a in A){
        u[s,a] <- R[s,a] + sum(P[s, ,a]*U[,t+1,1])

      }
    }
    U[,t,1] <- apply(u,FUN=max, MARGIN=1)
```

```
    U[,t,2] <- apply(u,FUN=which.max, MARGIN=1)
  }
  U[,,2]-1
}


############run 10 problems#############
#1
inventoryMDP(N=15, M=2, p = c(.5,.25,.25),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u)
{10*u})

#2
inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
             o.=function(u){ifelse(u==0,0,2*u+2)} ,
             g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u)
{10*u})

#3
inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
             o.=function(u){ifelse(u==0,0,1*u+5)} ,
             g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u)
{10*u})

#4
inventoryMDP(N=15, M=5, p = c(.25,.25,.25,.25,0,0 ),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {0}, h.=function(u) {4*u}, f.=function(u)
{15*u})

#5
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {0}, h.=function(u) {1*u}, f.=function(u)
{15*u})

#6
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {3*u}, h.=function(u) {1*u},
f.=function(u) {6*u})

#7
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {3*u}, h.=function(u) {0*u},
f.=function(u) {6*u})

#8
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1 ),
             o.=function(u){ifelse(u==0,0,1*u+2)} ,
             g.=function(u) {0*u}, h.=function(u) {0*u},
f.=function(u) {6*u})
```

```
#9
inventoryMDP(N=15, M=5, p = c(0.7,0.2,0.1,0,0,0 ),
             o.=function(u){ifelse(u==0,0,2*u+2)} ,
             g.=function(u) {5*u}, h.=function(u) {2*u},
f.=function(u) {30*u})


#10
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1),
             o.=function(u){ifelse(u==0,0,2*u)} ,
             g.=function(u) {0}, h.=function(u) {0.5*u},
f.=function(u) {6*u})

#PART b: different order cost
inventoryMDP(N=15, M=5, p = c(.1,.1,.3,.3,.1,.1),
             o.=function(u){ifelse(u>=3, 2*u - 5, u)} ,
             g.=function(u) {0}, h.=function(u) {u}, f.=function(u)
{10*u})
```

```
#######secretary.R - HW3 Q4#####
#set up
S <- c(1,2,3)
names(S) <- c("not best","best","stopped")
actions <- c("quit","continue")

P<- array(0,dim=c(3,3,2),dimnames=list(names(S),names(S),A))
P[ ,3,2] <- 1  #quit -> stopped
P[3,3,1] <- 1  #stopped -> stopped
P[1:2,1,1] <- 1/(t+1)
P[1:2,1,1] <- t/(t+1)

#backward induction
results <- array(0,
dim=c(3,50),dimnames=list(c("tau","u1(nb)","u1(b)"),c(1:50)))
results[ ,1] <- c(1,0,1)
for (N in 2:50) {
t <- N
U <- array(0,dim=c(3,N))
A <- array(100,dim=c(3,(N-1)))
U[2,t] <- 1   #h(not best) = h(stopped) = 0
for (t in (N-1):1){
  U[1,t] <- max(c(0, U[2,t+1]/(t+1) + U[1,t+1]*t/(t+1)))
  U[2,t] <- max(c(t/N, U[2,t+1]/(t+1) + U[1,t+1]*t/(t+1)))
  A[1,t] <- actions[which.max(c(0, U[2,t+1]/(t+1) +
U[1,t+1]*t/(t+1)))]
  A[2,t] <- actions[which.max(c(t/N, U[2,t+1]/(t+1) +
U[1,t+1]*t/(t+1)))]
  U[3,t] <- 0 #stopped, no reward
}
if (N == 2) {results[1,N] <- 1}
else {results[1,N] <- max(which(A[2,] == "continue"))}   #tau = index
of last continue
results[2:3,N] <- U[1:2,1]   #keep u*1
}
par(mfrow = c(1, 2))
#plot tau/N
plot(1:50,results[1,]/(1:50),type="l",ylim=c(0,1),xlab="Number of
candidates",ylab="Proportion to interview before selection")
points(1:50,results[1,]/(1:50),type="p",ylim=c(0,1),pch=3)
abline(1/exp(1),0, col="red",lty=2)
#plot value
plot(1:50,results[3,],type="l",ylim=c(0,1),xlab="Number of
candidates",ylab="Chance of choosing best candidate using optimal rule
(%)")
points(1:50,results[3,],type="p",ylim=c(0,1),pch=3)
abline(1/exp(1),0, col="red",lty=2)
```