X-2013-PSI-PT

```
> restart;
> n:=18;
```

$$n := 18$$

```
> deniveles:=array(0..n,[0,-1,2,0.1,1.9,-0.5,-1,1.5,0,-1,2,-1,-0.2
  ,0.3,-1.6,0.2,-0.9,-1.3,-2]);
```

deniveles := array(0 .. 18, [

(0) = 0
(1) = -1
(2) = 2
(3) = 0.1
(4) = 1.9
(5) = -0.5
(6) = -1
(7) = 1.5
(8) = 0
(9) = -1
(10) = 2
(11) = -1
(12) = -0.2
(13) = 0.3
(14) = -1.6
(15) = 0.2
(16) = -0.9
(17) = -1.3
(18) = -2

])

```
> hauteurs:=array(0..n);
```

hauteurs := array(0 .. 18, [ ])

```
> calculHauteurs:=proc(n)
  global deniveles,hauteurs;
  local i;
  hauteurs[0]:=0;
  for i from 1 to n do
       hauteurs[i]:=hauteurs[i-1]+deniveles[i]
  od;
  return ;
  end:
> calculHauteurs(18);
> eval(hauteurs);
```

array(0 .. 18, [

```
(0) = 0
(1) = -1
(2) = 1
(3) = 1.1
(4) = 3.0
(5) = 2.5
(6) = 1.5
(7) = 3.0
(8) = 3.0
(9) = 2.0
(10) = 4.0
(11) = 3.0
(12) = 2.8
(13) = 3.1
(14) = 1.5
(15) = 1.7
(16) = 0.8
(17) = -0.5
(18) = -2.5
])
```

```
> calculFenetre:=proc(n)
  global hauteurs,hMin,hMax,iMin,iMax;
  local i;
  hMin:=0;
  hMax:=0;
  iMin:=0;
  iMax:=0;
  for i from 1 to n do
       if hauteurs[i]<hMin then
            iMin:=i;
            hMin:=hauteurs[i]
       fi;
       if hauteurs[i]>hMax then
            iMax:=i;
            hMax:=hauteurs[i]
       fi;
  od;
  return ;
  end:
>
```

```
> calculFenetre(18);
```

```
> hMin,hMax,iMin,iMax;
```

-2.5, 4.0, 18, 10

```
> distanceAuSol:=proc(i,j)
  global deniveles,n;
  local k,distance;
  distance:=0;
  for k from i+1 to j do
       distance:=distance+sqrt(1+deniveles[k]*deniveles[k])
  od;
  return distance;
  end:
```

```
> distanceAuSol(0,1),evalf(distanceAuSol(0,18));
```

$\sqrt{2}$, 27.39386545

```
> estRemarquable:=proc(i)
  global hauteurs,n;
  local autour;
  if i=0 or i=n then
       return true
  else
       autour:=hauteurs[i-1];
       if hauteurs[i+1]>autour then
            autour:=hauteurs[i+1]
       fi;
       if autour<hauteurs[i] then
            return true
       else
            return false
       fi;
  fi;
  end:
```

```
> seq(estRemarquable(i),i=0..18);
```

true, false, false, false, true, false, false, false, false, false, true, false, false, true, false, true, false, false, true

```
> longueurDuPlusLongBassin:=proc(n)
  local iDebutBassin,k,longueurBassin;
  iDebutBassin:=0;
  longueurBassin:=0;
  for k from 1 to n do
       if estRemarquable(k) then
            if
  evalf(distanceAuSol(iDebutBassin,k)-longueurBassin)>0 then
                 longueurBassin:=distanceAuSol(iDebutBassin,k)
            fi;
            iDebutBassin:=k;print(k)
       fi;
  od;
```

```
  return longueurBassin;
  end:
> longueurDuPlusLongBassin(18);
```

4

10

13

15

18

$3.920809627 + 2\sqrt{2} + \sqrt{5}$

```
> distanceAuSol(4,10);
```

$3.920809627 + 2\sqrt{2} + \sqrt{5}$

```
> estDeltaAuDessusDuSol:=proc(i,j,l,Delta)
  global hauteurs;
  local k,beta;
  beta:=(hauteurs[j]-hauteurs[i])/(j-i);
  if i<j then
       for k from i+1 to j-1 do
            if (hauteurs[k]-hauteurs[i]+Delta-l)/(k-i)>beta then
                 return false
            fi;
       od;
       return true
  else
       return estDeltaAuDessusDuSol(j,i,l,Delta)
  fi;
  end:
> estDeltaAuDessusDuSol(0,1,2,0.5),estDeltaAuDessusDuSol(0,6,2,0.5
  ),estDeltaAuDessusDuSol(5,10,2,0.5),estDeltaAuDessusDuSol(5,14,2
  ,0.5),estDeltaAuDessusDuSol(14,16,2,0.5),estDeltaAuDessusDuSol(1
  4,18,2,0.5);
```

true, *false*, *true*, *false*, *true*, *true*

```
> poteaux:=array(0..n+1,[seq(-1,i=0..n+1)]);
```

poteaux := array(0 .. 19, [

(0) = -1

(1) = -1

(2) = -1

(3) = -1

(4) = -1

(5) = -1

(6) = -1

(7) = -1

(8) = -1

```
    (9) = -1
    (10) = -1
    (11) = -1
    (12) = -1
    (13) = -1
    (14) = -1
    (15) = -1
    (16) = -1
    (17) = -1
    (18) = -1
    (19) = -1
    ])
> placementGloutonEnAvant:=proc(n,l,Delta)
  global poteaux;
  local k,dernier;
  poteaux[0]:=1;
  poteaux[1]:=0;
  dernier:=0;
  for k from 2 to n do
       if estDeltaAuDessusDuSol(dernier,k,l,Delta)=false then
            poteaux[0]:=poteaux[0]+1;
            poteaux[poteaux[0]]:=k-1;
            dernier:=k-1
       fi;
  od;
  poteaux[0]:=poteaux[0]+1;
  poteaux[poteaux[0]]:=n;
  return ;
  end:
> placementGloutonEnAvant(18,2,0.5);
> eval(poteaux);
array(0 .. 19, [
    (0) = 4
    (1) = 0
    (2) = 5
    (3) = 13
    (4) = 18
    (5) = -1
    (6) = -1
    (7) = -1
    (8) = -1
    (9) = -1
```

```
    (10) = -1
    (11) = -1
    (12) = -1
    (13) = -1
    (14) = -1
    (15) = -1
    (16) = -1
    (17) = -1
    (18) = -1
    (19) = -1
    ])
> placementGloutonAuPlusLoin:=proc(n,l,Delta)
  global poteaux;
  local k,dernier;
  poteaux[0]:=1;
  poteaux[1]:=0;
  dernier:=0;
  while dernier<n do
       k:=n;
       while estDeltaAuDessusDuSol(dernier,k,l,Delta)=false do
            k:=k-1
       od;
       poteaux[0]:=poteaux[0]+1;
       poteaux[poteaux[0]]:=k;
       dernier:=k;
  od;
  return ;
  end:
> poteaux:=array(0..n+1,[seq(-1,i=0..n+1)]):
> placementGloutonAuPlusLoin(18,2,0.5);
> eval(poteaux);
array(0 .. 19, [
    (0) = 4
    (1) = 0
    (2) = 10
    (3) = 17
    (4) = 18
    (5) = -1
    (6) = -1
    (7) = -1
    (8) = -1
    (9) = -1
```

```
(10) = -1
(11) = -1
(12) = -1
(13) = -1
(14) = -1
(15) = -1
(16) = -1
(17) = -1
(18) = -1
(19) = -1
])
```

```
> optL:=array(0..n):precOptL:=array(0..n):
> longueurMinimale:=proc(n,l,Delta)
  global optL,deniveles,hauteurs,precOptL;
  local i,k,longueur;
  optL[0]:=0;
  precOptL[0]:=-1;
  for i from 1 to n do
       optL[i]:=optL[i-1]+sqrt(1+deniveles[i]*deniveles[i]);
       precOptL[i]:=i-1;
       for k from 2 to i do
            if estDeltaAuDessusDuSol(i-k,i,l,Delta) then

  longueur:=optL[i-k]+sqrt(k*k+(hauteurs[i]-hauteurs[i-k])*(hauteu
  rs[i]-hauteurs[i-k]));
                 if evalf(longueur-optL[i])<0 then
                      optL[i]:=longueur;
                      precOptL[i]:=i-k
                 fi;
            fi;
       od;
  od;
  return optL[n];
  end;
```

longueurMinimale := **proc**(*n*, *l*, Δ)

local *i*, *k*, *longueur*;

global *optL*, *deniveles*, *hauteurs*, *precOptL*;

optL[0] := 0;

precOptL[0] := -1;

for *i* **to** *n* **do**

optL[*i*] := *optL*[*i* – 1] + sqrt(1 + *deniveles*[*i*]^2);

precOptL[*i*] := *i* – 1;

```
            for k from 2 to i do
                if estDeltaAuDessusDuSol(i − k, i, l, Δ) then
                    longueur := optL[ i − k ] + sqrt( k^2 + ( hauteurs[ i ] − hauteurs[ i − k ] )^2 );
                    if evalf( longueur − optL[ i ] ) < 0 then
                        optL[ i ] := longueur; precOptL[ i ] := i − k
                    end if
                end if
            end do
        end do;
        return optL[ n ]
end proc
> longueurMinimale(18,2,0.5);
                                    20.38551643
> eval(optL);
array(0 .. 18, [
    (0) = 0
    (1) = √2
    (2) = √5
    (3) = 3.195309062
    (4) = 5.000000000
    (5) = √5 + 3.354101966
    (6) = √5 + 3.354101966 + √2
    (7) = 7.615773106
    (8) = 8.544003745
    (9) = √5 + 7.385230840
    (10) = 10.77032961
    (11) = 11.54400374
    (12) = 12.54900062
    (13) = 13.54500364
    (14) = 14.72866218
    (15) = 15.66369475
    (16) = 16.84762419
    (17) = 18.33421346
    (18) = 20.38551643
    ])
> eval(precOptL);
array(0 .. 18, [
    (0) = -1
    (1) = 0
```

```
    (2) = 0
    (3) = 0
    (4) = 0
    (5) = 2
    (6) = 5
    (7) = 0
    (8) = 0
    (9) = 5
    (10) = 0
    (11) = 8
    (12) = 8
    (13) = 8
    (14) = 8
    (15) = 8
    (16) = 14
    (17) = 14
    (18) = 14
    ])
> placementDesPoteaux:=proc(n)
  global poteaux,precOptL;
  local i,nbrePoteaux;
  nbrePoteaux:=1;
  poteaux[n+1]:=n;
  i:=precOptL[n];
  while i>-1 do
       poteaux[n+1-nbrePoteaux]:=i;
       nbrePoteaux:=nbrePoteaux+1;
       i:=precOptL[i]
  od;
  poteaux[0]:=nbrePoteaux;
  for i from 1 to nbrePoteaux do
       poteaux[i]:=poteaux[i+n+1-nbrePoteaux]
  od;
  for i from nbrePoteaux+1 to n+1 do
       poteaux[i]:=-1
  od;
  return ;
  end:
> poteaux:=array(0..n+1,[seq(-1,i=0..n+1)]):
> placementDesPoteaux(18);
> eval(poteaux);
array(0 .. 19, [
```

```
(0) = 4
(1) = 0
(2) = 8
(3) = 14
(4) = 18
(5) = -1
(6) = -1
(7) = -1
(8) = -1
(9) = -1
(10) = -1
(11) = -1
(12) = -1
(13) = -1
(14) = -1
(15) = -1
(16) = -1
(17) = -1
(18) = -1
(19) = -1
])
>
```