Vistula, IT Faculty, 2014

Algorithms and Complexity

Dmitry A. Zaitsev
http://daze.ho.ua

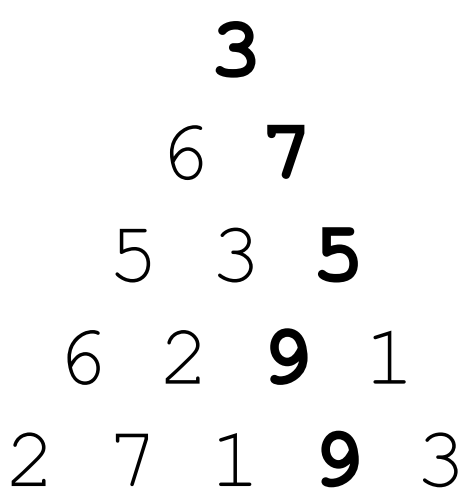
Lecture 2:

A case study:
paths in a triangle of numbers

A task an its intuitive solutions

Rules:
go down
only two ways –
one left or one right

Goal:
maximal sum

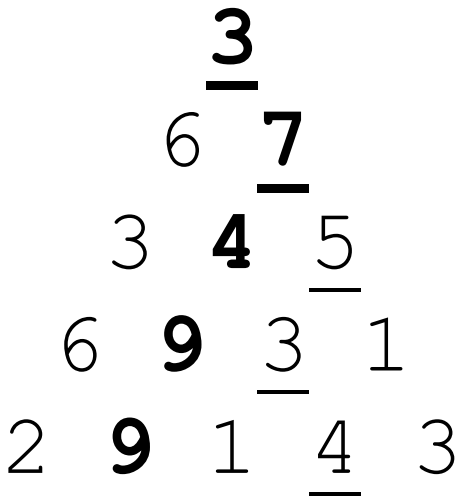


I. Greedy approach – at each choice take the greater of two numbers

II. Exhaustive search – take the best of all the variants

3+6+5+6+2=22
3+6+5+6+7=27
3+6+5+2+7=23
3+6+5+2+1=17
3+6+3+2+7=21
3+6+3+2+1=15
3+6+3+9+1=22
3+6+3+9+9=30
3+7+3+2+7=22
3+7+3+2+1=16
3+7+3+9+1=23
3+7+3+9+9=31
3+7+5+9+1=25
3+7+5+9+9=33
3+7+5+1+9=25
3+7+5+1+3=19

Generally greedy algorithm does not give maximal sum



Optimal solution

3+7+4+9+9=32

Greedy solution

3+7+5+3+4=22

But it is very simple! Thus, it could be useful.

Data structures and rules

Half a matrix to represent the triangle – a[n][n]

i \ j	0	1	2	3	4
0	**3**				
1	**6**	**7**			
2	**3**	**4**	**5**		
3	**6**	**9**	**3**	**1**	
4	**2**	**9**	**1**	**4**	**3**

n=5

Indices range

0 <= i < n,
0 <= j <= i

Neighbors of a[i][j]

a[i+1][i]
a[i+1][i+1]

Greedy algorithm

```
ip=0;
bms=a[0][0];
printf("\ngreedy path:");
printf("%d ",ip);

for(i=1; i<n; i++)
{
  if(a[i][ip] >= a[i][ip+1]) ip=ip; else ip=ip+1;
  bms+=a[i][ip];
  printf("%d ",ip);
}
printf("\n");
printf("\nsum=%ld\n", bms);
```

Exhaustive search idea – from the current to the next

An index of a current path – (p[0],p[1],…,p[n-1])

i \ j	0	1	2	3	4
0	**3**				
1	**6**	**7**			
2	**3**	**4**	**5**		
3	**6**	**9**	**3**	**1**	
4	**2**	**9**	**1**	**4**	**3**

i	p[i]
0	0
1	1
2	1
3	2
4	3

- Start from p=(0,0,…,0)
- Move the last index p[i]=p[i]+1
- If not possible, when p[i]=p[i-1]+1,
  then move the previous p[i-1]

Exhaustive search algorithm

```
bms=0;
for(i=0;i<n;i++){p[i]=0;bp[i]=0;bms+=a[i][0];}

go=1;
while(go)
{

 //next
 go=0;
 for(i=n-1;i>0;i--)
 {
  if(p[i]<i && p[i]<=p[i-1])
  {
          p[i]++;
          for(k=i+1;k<n;k++)p[k]=p[k-1];
          go=1;
          break;
  }
 }

 if(!go) break;

 //calculate sum
 ms=0;
 for(i=0;i<n;i++){ms+=a[i][p[i]];}

//copy the best
 if(ms>bms)
 {
  for(i=0;i<n;i++){bp[i]=p[i];}
  bms=ms;
 }

}

printf("\nbms=%ld\n", bms);
printf("best path:");
for(i=0;i<n;i++){printf("%d ",bp[i]);}
printf("\n");
```

Exhaustive search gives a little more than nothing

Generate triangles with random numbers and try them

n	Time of work
5	< 1 s
10	< 1 s
20	< 1 s
30	~ 1 min.
40	?

Dynamic programming – store and use intermediate results

i \ j	0	1	2	3	4
0	**3**				
1	**6**	**7**			
2	**3**	**4**	**5**		
3	**6**	**9**	**3**	**1**	
4	**2**	**9**	**1**	**4**	**3**

B[i][j]:

If(j==0)
b[i][j]=a[i][j]+b[i-1][j];

If(j==i)
b[i][j]=a[i][j]+b[i-1][j-1]

else
max(
(a[i][j]+b[i-1][j-1]),
(a[i][j]+b[i-1][j]))

Matrix of the best sums – b[n][n]

i \ j	0	1	2	3	4
0	**3**				
1	**9**	**10**			
2	**12**	**14**	**15**		
3	**18**	**23**	**18**	**16**	
4	**20**	**32**	**24**	**22**	**19**

Dynamic programming algorithm

```
// calculate matrix b
b[0][0]=a[0][0];
i=1;j=0;
while(i<n)
{
  if(j==0) b[i][j]=a[i][j]+b[i-1][j]; else
    if(j==i) b[i][j]=a[i][j]+b[i-1][j-1]; else
      b[i][j]=max((a[i][j]+b[i-1][j-1]),
                  (a[i][j]+b[i-1][j]));

  if(++j>i){i++;j=0;}
}

// find maximal in the last row of b
ms=b[n-1][0];
jm=0;
for(j=1;j<n;j++)
{
  if(b[n-1][j]>ms)
   { ms=b[n-1][j]; jm=j; }
}
```

```
// recover the path
p[n-1]=jm;
for(i=n-1;i>=1;i--)
{
  if(p[i]==0) p[i-1]=0;
   else if(p[i]==i) p[i-1]=i-1;
          else
          {
            pr=b[i][p[i]];
            if(pr==a[i][p[i]]+b[i-1][p[i]-1])
              p[i-1]=p[i]-1;
            else
              p[i-1]=p[i];
          }
}

// print results
printf("sum=%ld\nmax patk: ", ms);
for(i=0;i<n;i++) printf("%d ",p[i]);
printf("\n");
```

Dynamic programming algorithm works for big triangles

n	Time of work for Exhaustive search	Time of work for Dynamic programming
5	< 1 s	< 1 s
10	< 1 s	< 1 s
20	< 1 s	< 1 s
30	~ 1 min	< 1 s
40	?	< 1 s
1000	?	~ 1 s

Auxiliary algorithms

Read a triangle

```
scanf("%d",&n);
i=0;j=0;
while(i<n)
{
  scanf("%d",&x);
  a[i][j++] = x;
  if(j>i){i++;j=0;}
}
```

Print a triangle

```
i=0;j=0;
while(i<n)
{
  printf("%d ",a[i][j++]);
  if(j>i){i++;j=0;printf("\n");}
}
```

Generate a triangle

```
int main(int argc, char *argv[])
{
  int n,i,j,r;
  if(argc<2) return;
  n=atoi(argv[1]);
  printf("%d\n",n);
  i=0;j=0;
  while(i<n)
  {
    r=rand()%100;
    printf("%d\n",r);
    j++;
    if(j>i){i++;j=0;}
  }
}
```