Технології розробки алгоритмів розв'язання інженерних задач

Лекція №6

Викладач: Дрєєв Олександр Миколайович

6. Рекурсивні алгоритми.
6.1. Задача програмного множення.
6.2. Пошук найбільшого спільного дільника.
6.3. Хвильовий алгоритм на базі рекурсивного.
6.4. Кількість варіантів шляхів «черепахи».

Рекурентні алгоритми

Що таке “рекурсія”

Рекурсивний алгоритм — алгоритм, який спрощує задачу або розбиває її на простіші і викликає себе для розв’язання цих спрощених підзадач. Обов’язкова присутність умови припинення подальших викликів.

Приклад: Множення m*n. Можна записати: m*n=(m-1)*n+n.

Алгоритм “множу(m,n)”: якщо m>1 повертаємо “множу(m-1,n)”+n; інакше повертаємо n.

Рекурентні алгоритми

Що таке “рекурсія”

Опис роботи “множу”:

“множу(3,8)”: якщо 3>1
повертаємо “множу(3-1,8)”+8;
інакше повертаємо 8.

“множу(2,8)”: якщо 2>1
повертаємо “множу(2-1,8)”+8;
інакше повертаємо 8.

“множу(1,8)”: якщо 1>1
повертаємо “множу(1-1,8)”+8;
інакше повертаємо 8.

Жадібні алгоритми

Що таке “рекурсія”

Схема роботи “множу”:

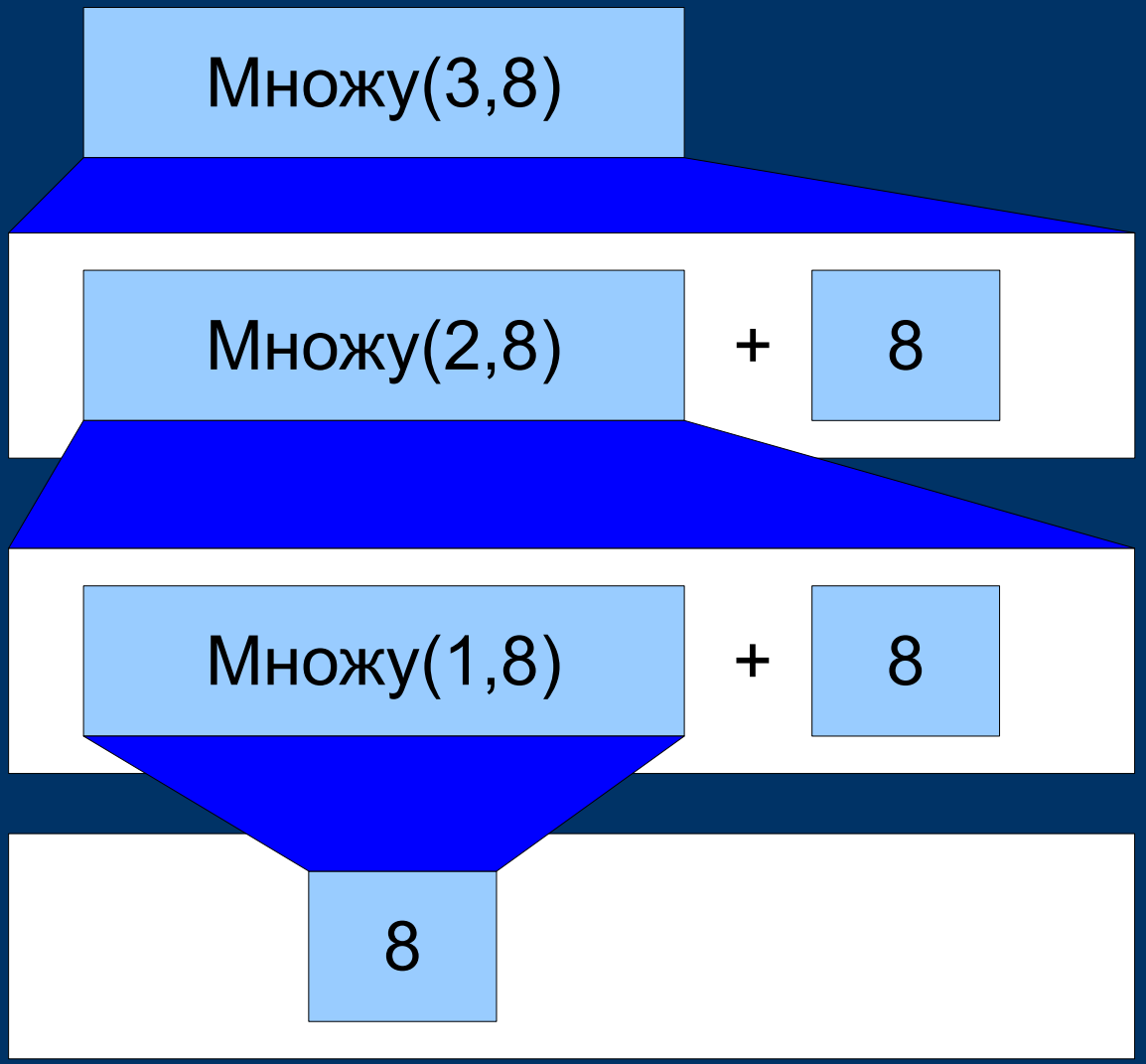
Задача 3*8 → 8 + простіша задача 2*8

Задача 2*8 → 8 + простіша задача 1*8

Задача 1*8 → 8

Рекурентні алгоритми

Що таке “рекурсія”



Рекурентні алгоритми

Характеристика рекурсії

Переваги рекурсії:

1. Значне спрощення запису алгоритмів певного класу задач.
2. Відкрита логіка роботи.

Недоліки:

1. Обмежена глибина викликів.
2. Витрати на виклик функції суттєво сповільнюють роботу.
3. Іноді важко написати варіант програми без рекурентного виклику.

Рекурентні алгоритми

Пошук найбільшого спільного дільника

Задача: знайти K — спільний найбільший дільник чисел M > N.

Спрощена задача: знайти K — спільний найбільший дільник чисел (M-N), N.

Алгоритм: Більше число називаємо M, менше N.
Якщо M=N, тоді повертаємо K=N;
інакше K=Алгоритм(M-N,N).

Рекурентні алгоритми

Пошук найбільшого спільного дільника

Реалізація:

```
int NSD(int M, int N);
int main()
{  int M=345, N=875;
   int K=NSD(M,N);
}
int NSD(int M, int N)
{ if( N==M ) return N;
  if( N>M){ int T=M; M=N; N=T; }
  return NSD(M-N,N);
}
```

Рекурентні алгоритми

Пошук найбільшого спільного дільника

Реалізація:

```
int NSD(int M, int N);
int main()
{  int M=345, N=875;
   int K=NSD(M,N);
}
int NSD(int M, int N)
{ if( N==M ) return N;
  if( N>M){ int T=M; M=N; N=T; }
  return NSD(M-N,N);
}
```

Рекурентні алгоритми

Пошук найбільшого спільного дільника

865	345
520	345
175	345
170	175
170	5

865-345

520-345

345-175

175-170

Рекурентні алгоритми

Пошук коротшого шляху, хвильовий алгоритм

Пошук найкоротшого шляху:

0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	5	6		8	9	10	11
2								9	9	10	11
3	3	4	5	6	7	8					11
4	4	4	5	6	7	8		14	13	12	12
					7	8		14	13	13	13
12	11	10	9	8	8	8		14			
12	11	10	9	9	9	9		15	15	16	17
12	11							16	16	16	17
12	12	12	13	14	15	16					17
13	13	13	13	14	15	16	17	18	19	18	18
14	14	14	14	14	15	16	17	18	19	19	19

Рекурентні алгоритми

Пошук коротшого шляху, хвильовий алгоритм

Аналіз задачі, ідея:

1. Якщо я знаю кількість кроків для досягнення своєї клітинки, досягаю наступним кроком сусідні клітинки.
2. Якщо в сусідній клітинці вже є число, менше за моє+1, то туди вже є коротший шлях.
3. Якщо в сусідній клітинці число більше за моє, то потрібно його замінити своїм, то туди є коротший шлях.

Рекурентні алгоритми

Пошук коротшого шляху, хвильовий алгоритм

Алгоритм КРОК(N):

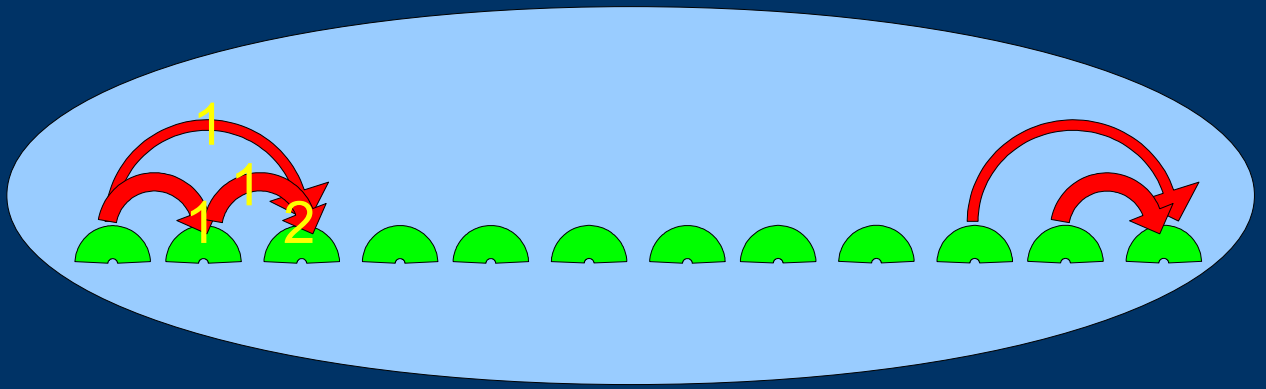
1. Число кроків N (на початку 0).
2. Якщо в сусідній клітинці вже є число, менше за N+1, то туди вже є коротший шлях.
3. Якщо в сусідній клітинці число більше за моє, або там немає числа то КРОК(N+1).

Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Задача:

Через болото ведуть кочки. Коник може стрибнути через кочку або на наступну. Знайдіть кількість варіантів шляху коника через болото.

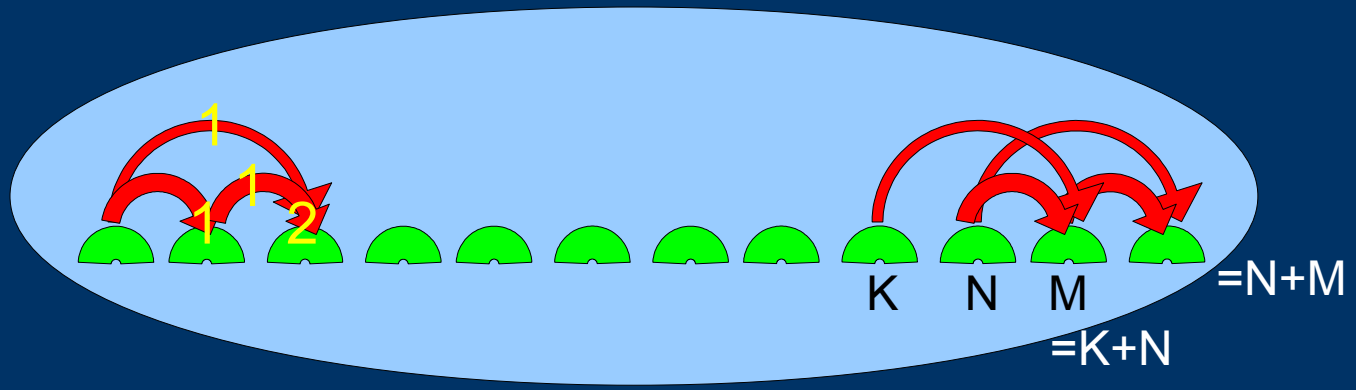


Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Аналіз:

Якщо відомо кількість шляхів до двох попередніх кочок, то кількість шляхів буде сумою цих варіантів.



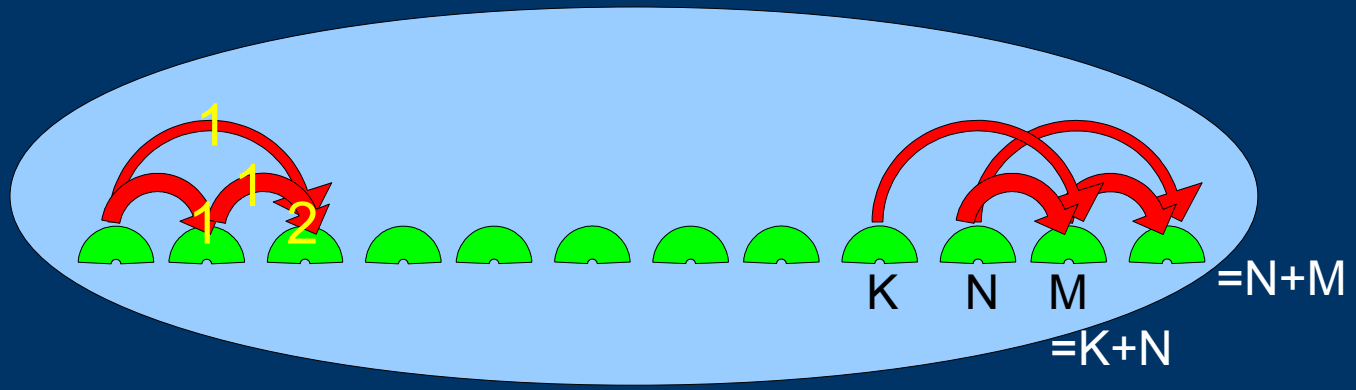
Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Алгоритм:

Варіанти(0)=1; Варіанти(1)=1;

Варіанти(N)=Варіанти(N-1)+Варіанти(N-2).



Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Складність алгоритму:

Алгоритм прослідкує всі варіанти, а при N+1 кількість варіантів майже подвоюється — експонентна складність! Для N=100 розрахунки триватимуть 2^{100} мікросекунд, або декілька міліардів років!

Висновок: в такому вигляді алгоритм не застосовний.

Рекурентні алгоритми
Динамічне програмування

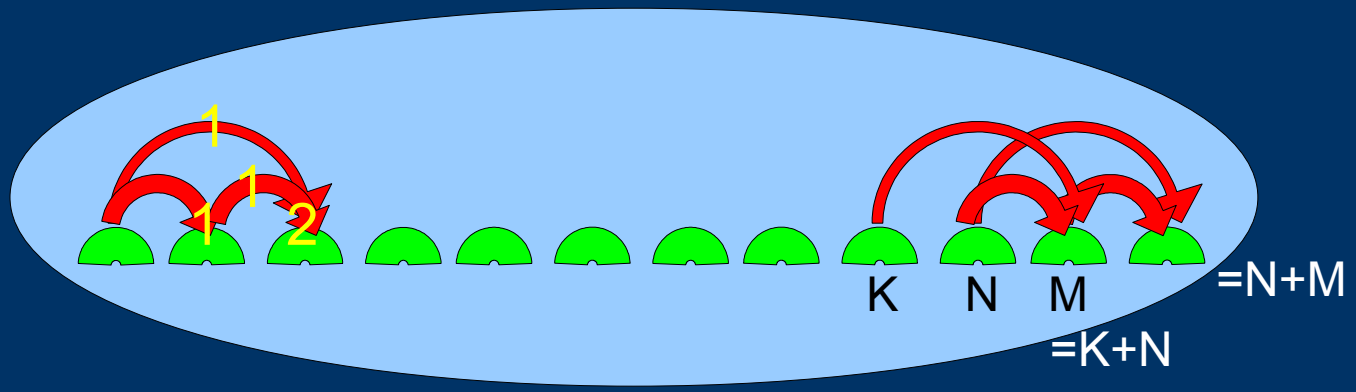
Шлях коника:

Розрахунок:

Варіанти(M+1)=Варіанти(M)+Варіанти(N).

Варіанти(M) =Варіанти(N) +Варіанти(K).

Висновок: для більшості кочок функція викликається багато раз.



Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Алгоритм:

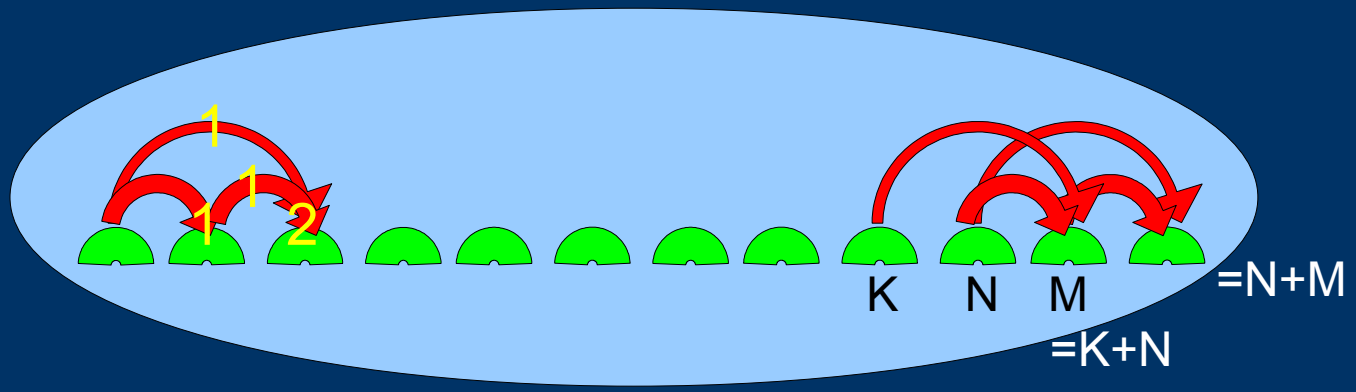
Масив:

1	1	2	...	F(N-2)	F(N-1)	F(N)

Варіанти(N)=Варіанти(N-1)+Варіанти(N-2).

Якщо в таблиці немає Варіанти(N-1), тоді

Варіанти(N-1)=Варіанти(N-2)+Варіанти(N-3).



Рекурентні алгоритми
Динамічне програмування

Шлях коника:

Алгоритм:

Масив:

1	1	2	...	F(N-2)	F(N-1)	F(N)

Варіанти(N)=Варіанти(N-1)+Варіанти(N-2).

Якщо в таблиці немає Варіанти(N-1), тоді

Варіанти(N-1)=Варіанти(N-2)+Варіанти(N-3).

Для кожного числа в таблиці розрахунок робиться один раз. Алгоритм ЛІНІЙНИЙ.

Рекурентні алгоритми
Динамічне програмування

Динамічне програмування — рекурсивний алгоритм в якому деякі варіанти спрощених задач повторюються, і замість їх повторного розрахування використовують готову відповідь яку знайдено раніше.

Переваги:

Має меншу складність

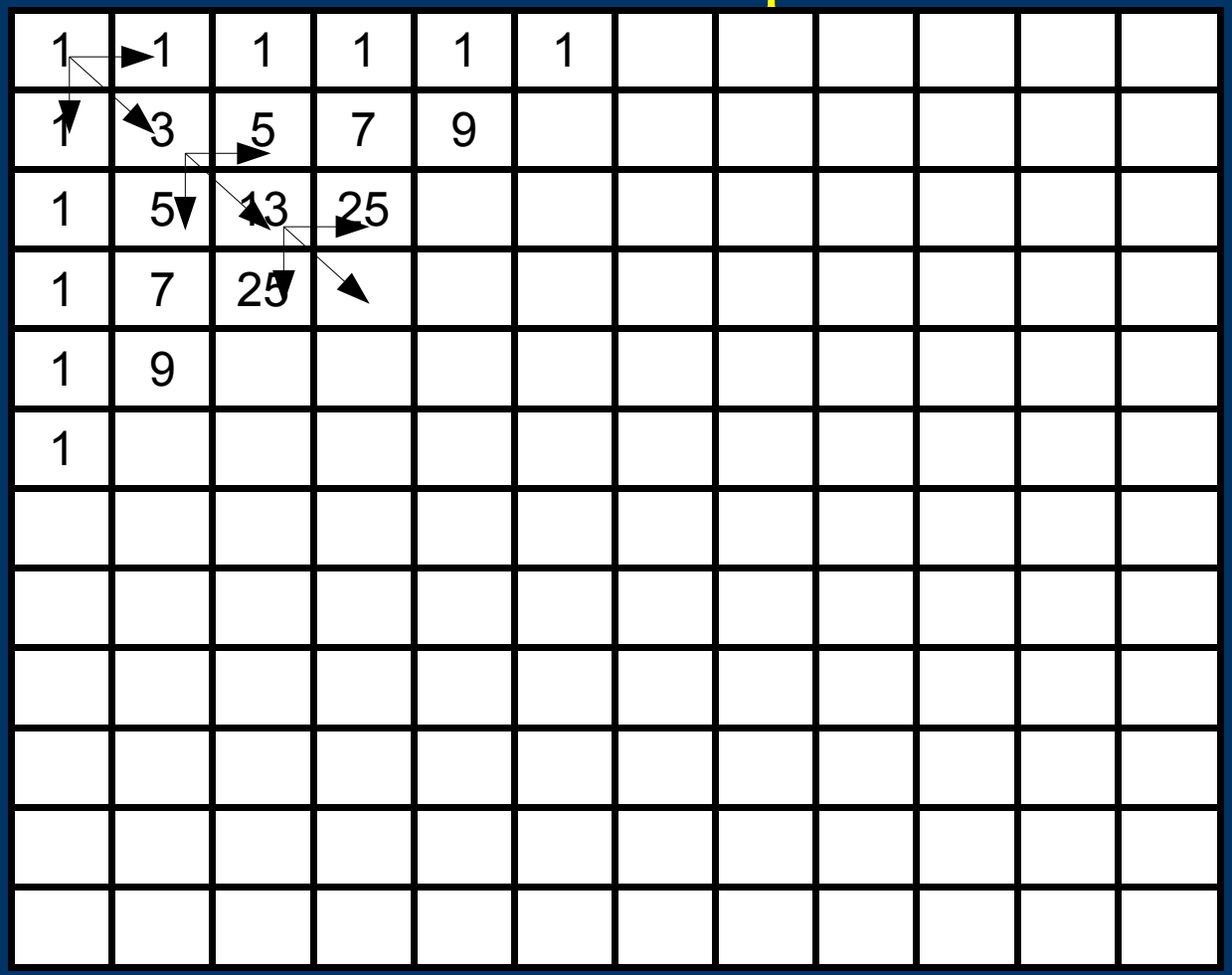
Недоліки:

Вимагає додаткову пам’ять для проміжних результатів.

Рекурентні алгоритми

Динамічне програмування

Задача на кількість шляхів черепахи:



Рекурентні алгоритми
Динамічне програмування

Задача на кількість шляхів черепахи.

Рекурентний алгоритм:

Для клітки x,y:

Варіантів(x,y)=Варіантів(x-1,y)+Варіантів(x,y-1)
+Варіантів(x-1,y-1)

Варіантів(0,0)=1.

Кількість викликів для однієї клітини?

Складність 3^{nm}

(n,m) — розмір поля

Рекурентні алгоритми

Динамічне програмування

Задача на кількість шляхів черепахи.

Динамічний алгоритм:

Для клітки x,y:

Якщо (x,y) є в таблиці — вертаємо його. Інакше

Варіантів(x,y)=Варіантів(x-1,y)+Варіантів(x,y-1)
+Варіантів(x-1,y-1)

Варіантів(0,0)=1.

Кількість викликів для однієї клітини 1.

Складність n*m

(n,m) — розмір поля

Рекурентні алгоритми
Динамічне програмування

Задача на кількість шляхів черепахи.

Реалізація:

```
int tabl[10][10];
int variant(int x, int y);

int main()
{
   printf("%d\n",variant(9,9));
   return 0;
}
```

Рекурентні алгоритми
Динамічне програмування

Задача на кількість шляхів черепахи.

Реалізація:

```
//int tabl[10][10];
int variant(int x, int y)
{ int n=0;
   if(tabl[x][y]>0) return tabl[x][y];
   if( x>0 ) n+=variant(x-1,y);
   if( y>0 ) n+=variant(x,y-1);
   if( x>0 && y>0 ) n+=variant(x-1,y-1);
   tabl[x][y] = n;       return n;
}
```