Funzioni ricorsive e backtracking

Palestra di algoritmi

2020-2021

Ricorsione

Motivo

Il modo più 'semplice' di risolvere alcuni problemi è tramite funzioni ricorsive. Una funzione ricorsiva ha:

- una parte 'operativa' in cui invoca sè stessa una o più volte **con un input di dimensione minore**.
- uno o più casi base che indicano quando ci si deve fermare.

Alcuni esempi

- Definizioni matematiche ricorsive
- Algoritmi di ordinamento (MergeSort)
- Backtracking

Backtracking

Principio

In alcuni problemi è richiesto o necessario esplorare l'intero spazio delle soluzioni ammissibili.

- Viene richiesto di elencare tutte le soluzioni possibili
- Non esiste una soluzione efficiente per risolvere un problema

Idea: *Prova a fare qualcosa; se non va bene, disfalo e prova qualcos'altro*

Esempio

N regine: piazzare N regine su una scacchiera $N \times N$ in modo che non si minaccino

- Video 1
- Video 2

Domino massimale

Problema

Sono date N tessere di domino, dove ogni tessera contiene due numeri compresi tra 0 e 6. Le tessere possono essere ruotate e la regola impone che due tessere possono essere concatenate se le loro estremità in contatto hanno inciso lo stesso numero. Trovare il maggior numero di tessere che si possono concatenare a formare un'unica catena.

Nota

In generale, più configurazioni possono soddisfare i requisti del problema: è sufficiente fornire la lunghezza massima.

Domino massimale

Idea

- Notiamo che $N \leq 10$, quindi possiamo permetterci una soluzione 'poco efficiente'
- Possiamo provare tutte le possibili permutazioni di tessere
- Le permutazioni di N elementi sono
  $N * (N-1) * (N-2) * \cdots * 2 * 1 = N!$
- Però ogni tessera possiamo girarla in due direzioni:

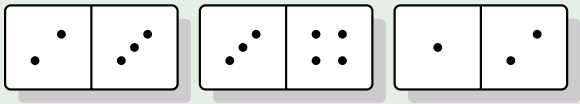
$$2N * 2(N-1) * 2(N-2) * \cdots * (2*2) * (2*1) = 2^N * N!$$

$$2^{10} * 10! = 1024 * 3628800 \approx 3.7 * 10^9$$

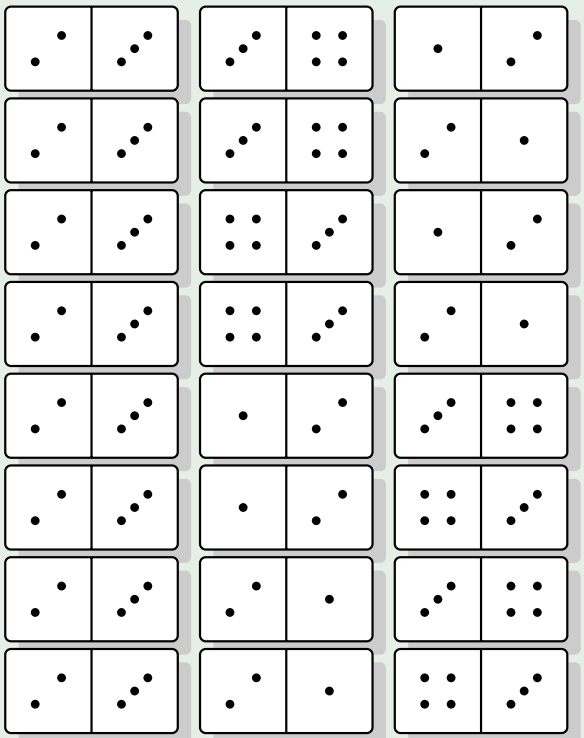
- Sono tantissime ma la maggior parte possiamo scartarle!

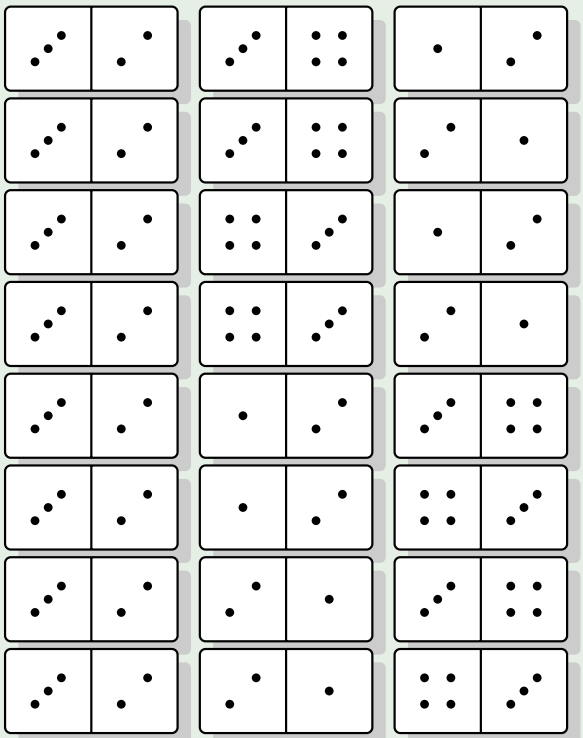
Domino massimale

Esempio

$N = 3$ 

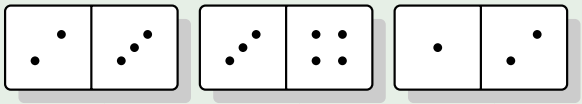
Permutazioni $= 2^3 * 3! = 8 * 6 = 48$



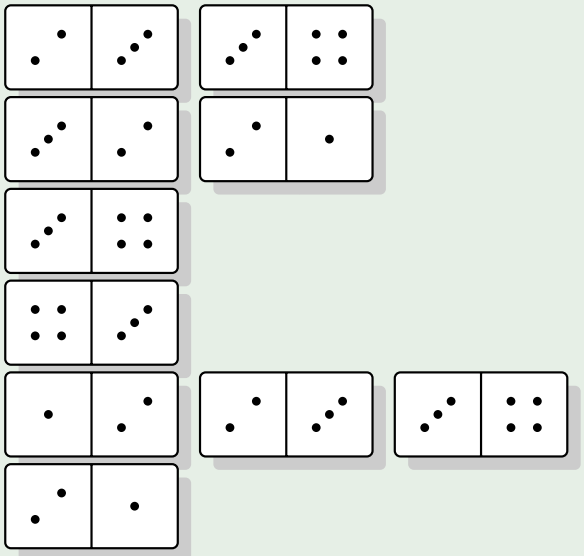


Domino massimale

Esempio

$N = 3$ 

Soluzioni valide $= 6$

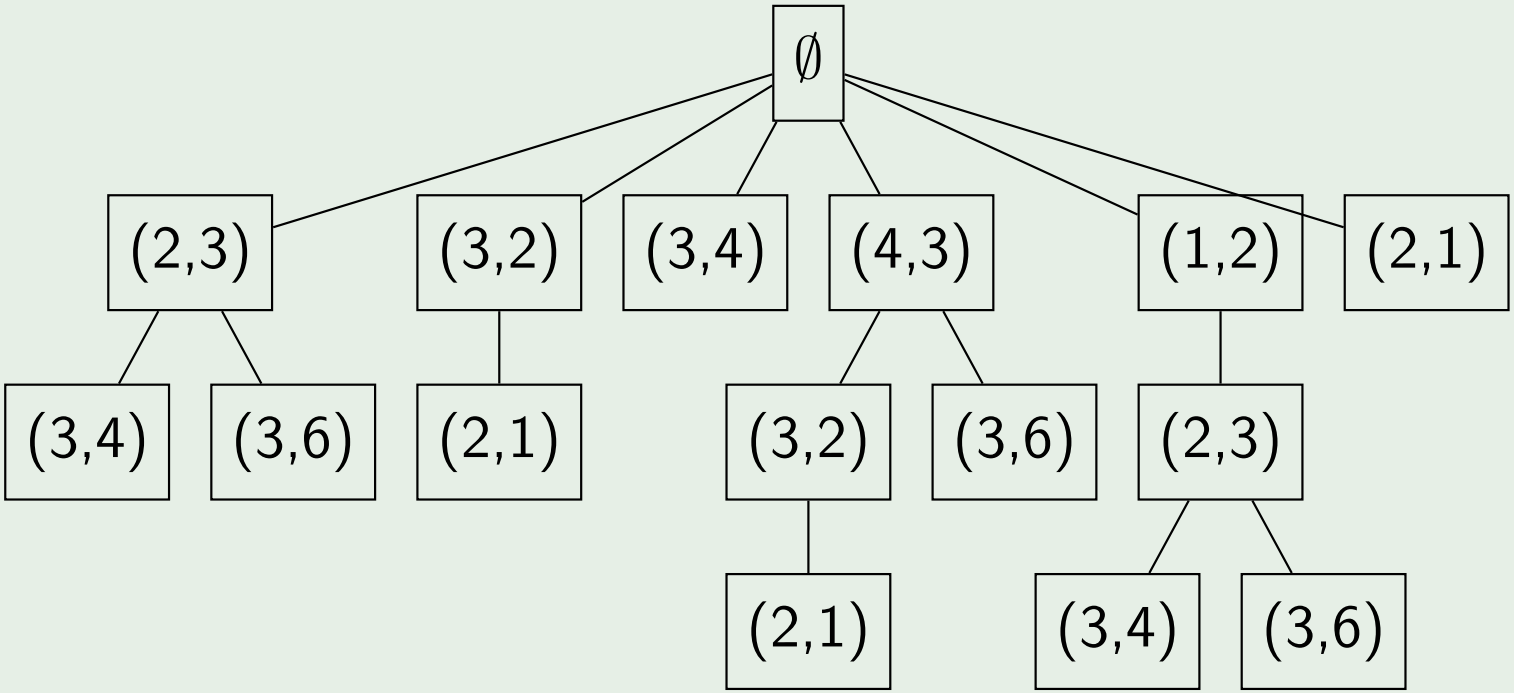


Domino massimale

Altro esempio

$N = 4$, tessere: (2, 3) (3, 4) (1, 2) (6, 3)

Possiamo visualizzare le soluzioni in un 'albero'!



La catena più lunga è la 'profondità' dell'albero.

Soluzione

Pseudocodice

In questi casi il metodo più 'semplice' è scrivere una soluzione ricorsiva

```
/** last indica il valore dell'ultimo elemento della catena corrente **/
int domino_rec(int last){
  /** n_tiles = 0 **
  /** Per ogni tessera che non ho ancora usato **/
  /**     Se posso metterla 'dritta' **/
  /**         Segno che l'ho usata **/
  /**         l = 'coda' della tessera **/
  /**         len = 1 + domino_rec(l) **/
  /**         n_tiles = max(n_tiles, len) **/
  /**         Segno che non l'ho usata **/
  /**     Se posso metterla 'ruotata' **/
  /**         Segno che l'ho usata **/
  /**         l = 'testa' della tessera **/
  /**         len = 1 + domino_rec(l) **/
  /**         n_tiles = max(n_tiles, len) **/
  /**         Segno che non l'ho usata **/
  /** Ritorno n_tiles **/
}
```

Soluzione

Codice

Possiamo tradurlo in codice C++

```
struct tile {
    int a, b;
};
vector<tile> t; // vettore di tessere
vector<bool> used; // used[i] = true -> ho usato la tessera i
/** last indica il valore dell'ultimo elemento della catena corrente **/
int domino_rec(int last){
    int n_tiles = 0, // lunghezza massima che ha la catena da quì in poi
        len;
    for(int i = 0; i < t.size(); i++){ // provo tutte le tessere
        if(!used[i] && last == t[i].a){ // se non ho ancora usato la tessera i e posso usarla
            used[i] = true; // la uso
            len = 1 + domino_rec(t[i].b); // continuo la catena ricorsivamente
            n_tiles = max(n_tiles, len); // tengo la lunghezza massima
            used[i] = false; // questa tessera è riutilizzabile
        }
        if(!used[i] && last == t[i].b){ // provo a usarla ruotata
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

Unendo tutto

```cpp
#include <fstream>
#include <vector>
using namespace std;
struct tile {
    int a, b;
};
vector<tile> t;
vector<bool> used;
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

```cpp
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int main() {
    ifstream in("input.txt");
    ofstream out("output.txt");
    int N;
    in >> N;
    for(int i = 0; i < N; i++){
        tile x;
        in >> x.a >> x.b;
        t.push_back(x);
        used.push_back(false);
    }
    out << domino() << endl;
    return 0;
}
```

Cosa succede?

Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

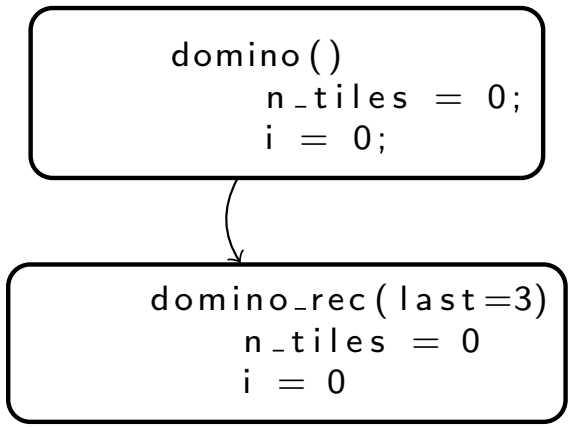
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = { false,  false,  false,  false};
```

```
domino()
    n_tiles = 0;
    i = 0;
```

Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

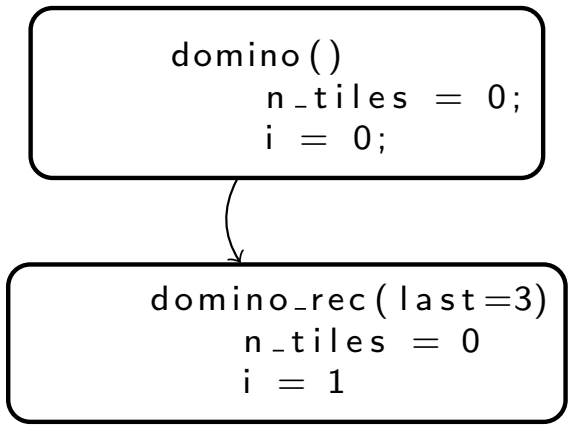
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

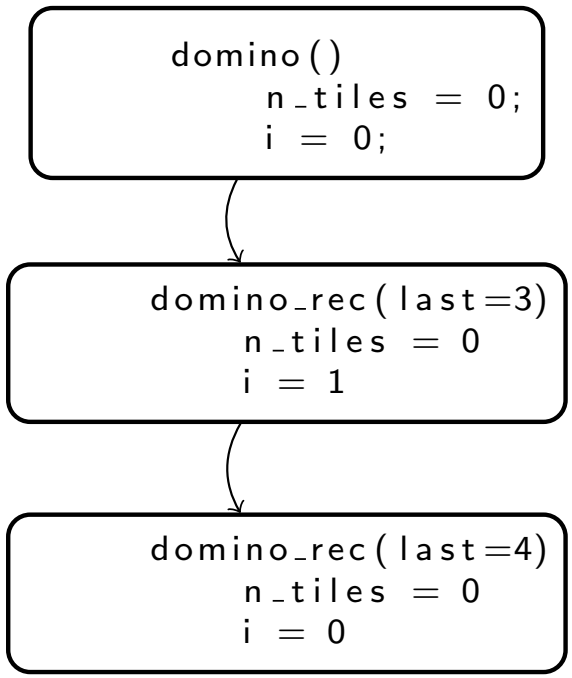
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

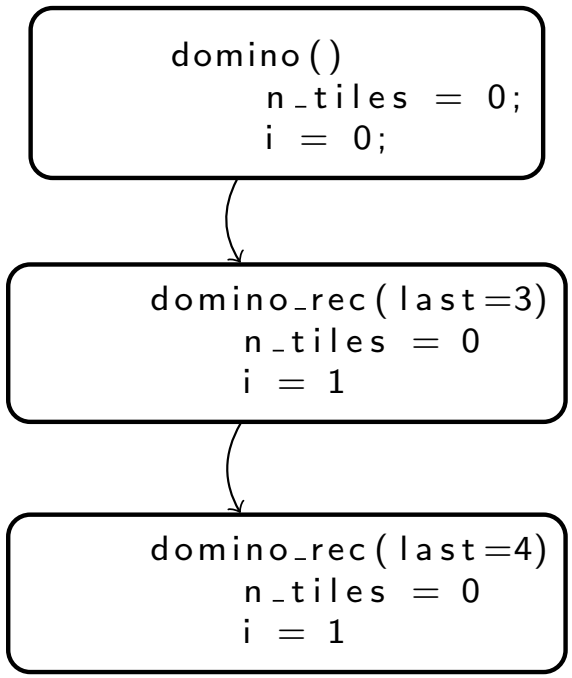
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,   true,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

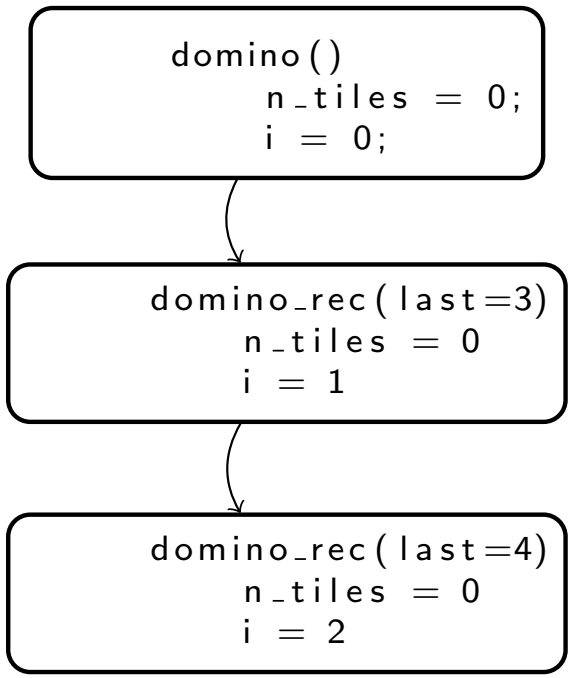
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,   true,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

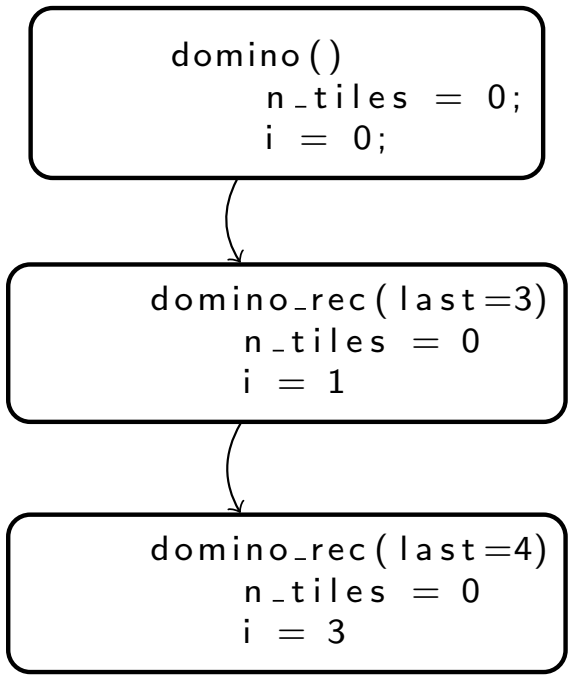
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,   true,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

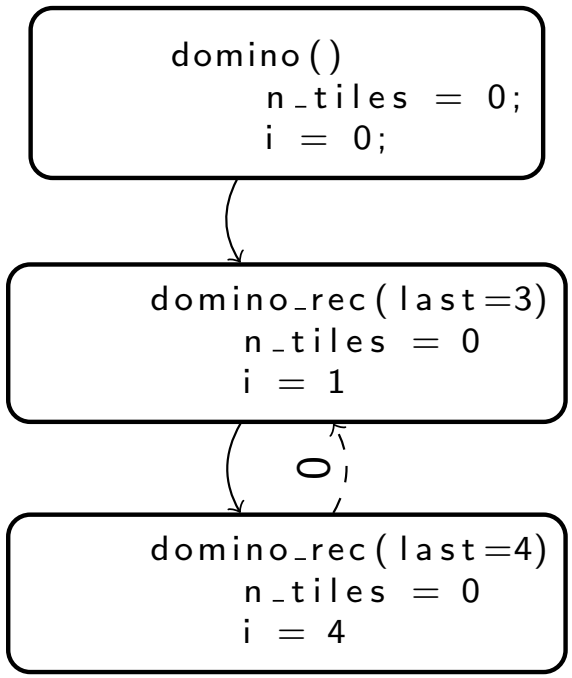
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,   true,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

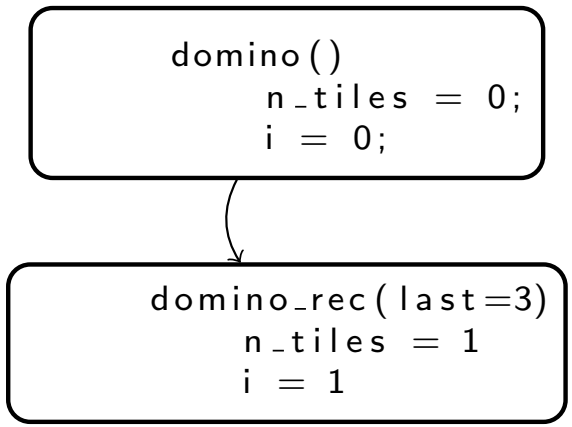
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,   true,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

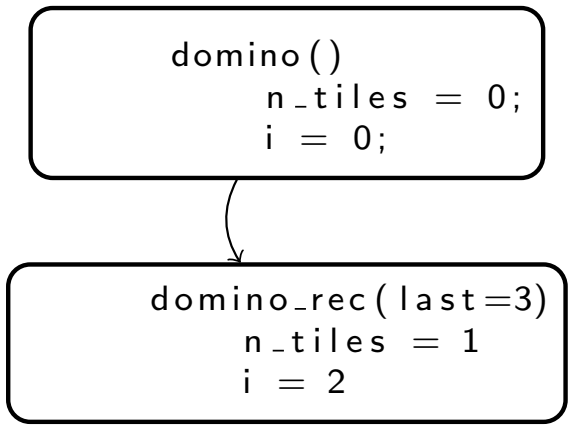
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

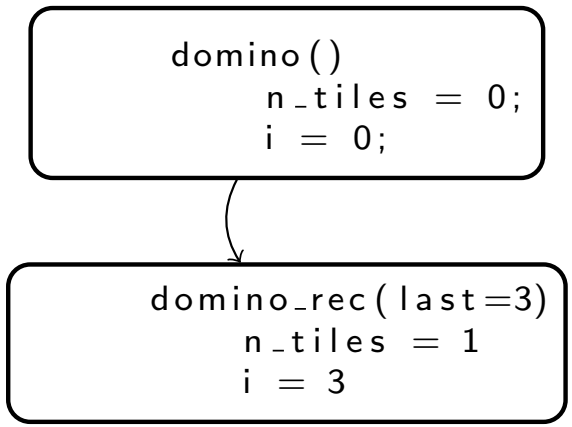
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

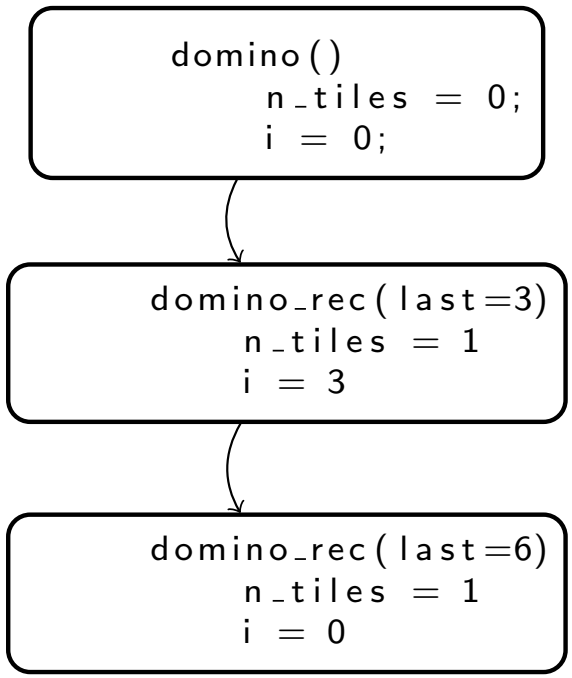
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

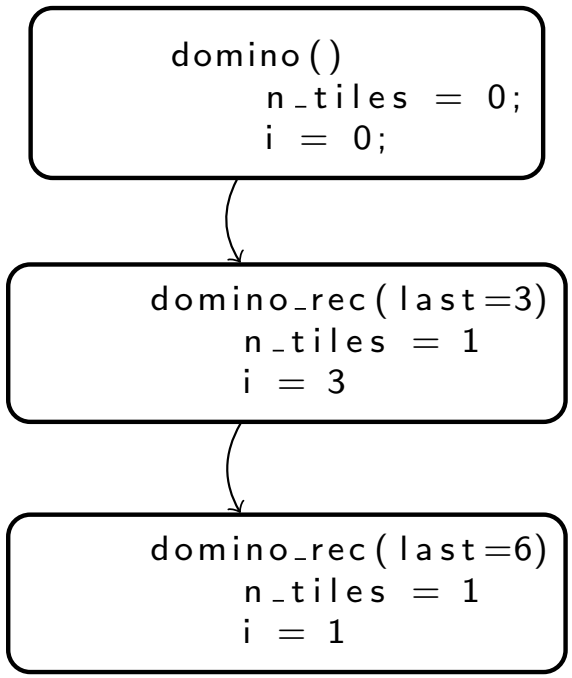
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,   true,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

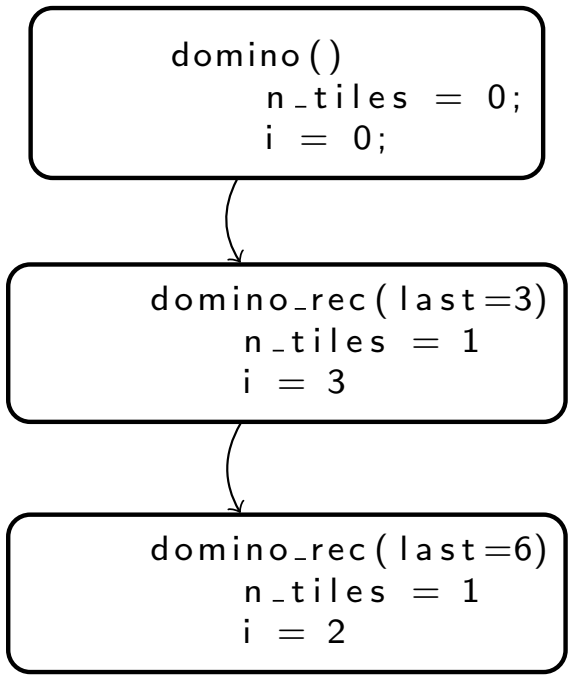
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,   true,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

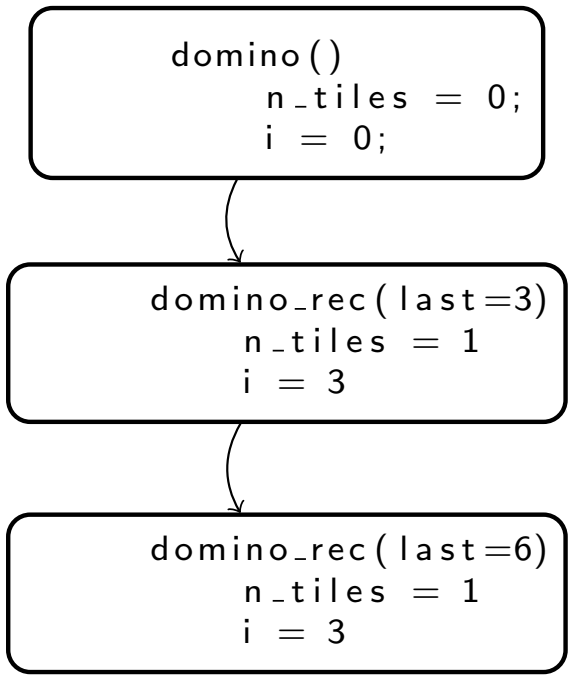
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,   true,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

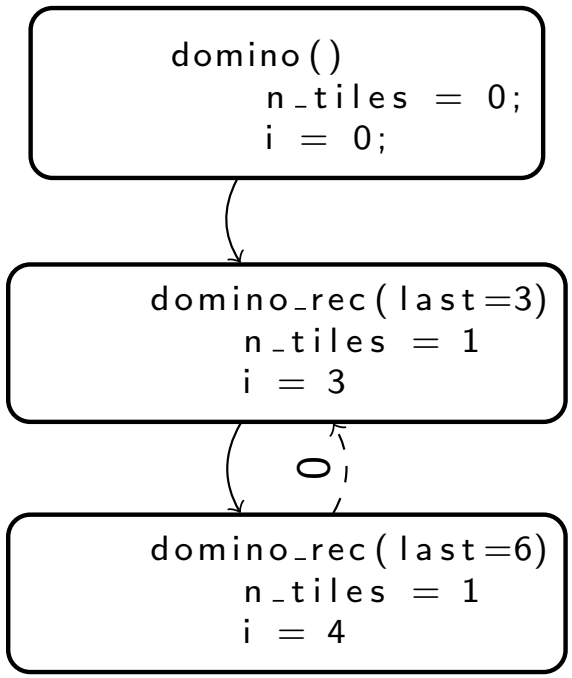
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,   true,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

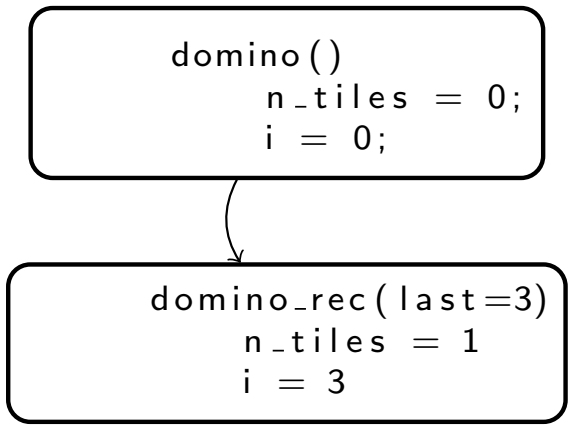
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,   true,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

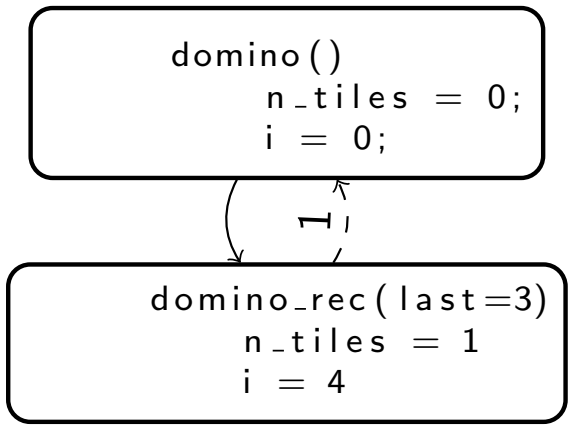
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  true,  false,  false,  false};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

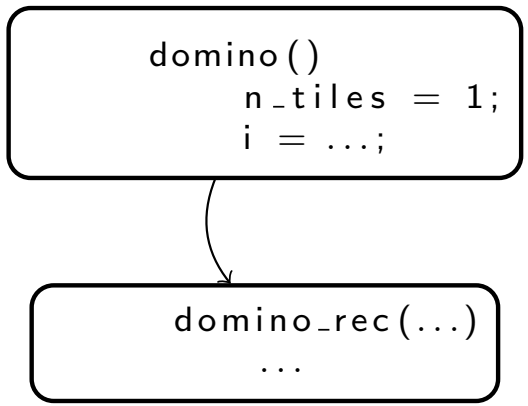
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {   ...,   ...,   ...,   ...};
```

```
domino()
    n_tiles = 1;
    i = ...;
```

Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

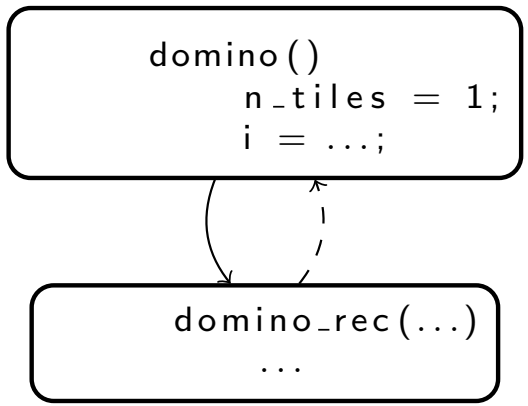
```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {   ...,   ...,   ...,   ...};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {   ...,   ...,   ...,   ...};
```



Cosa succede?

```
int domino(){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        used[i] = true;
        len = 1 + domino_rec(t[i].b);
        n_tiles = max(n_tiles, len);
        len = 1 + domino_rec(t[i].a);
        n_tiles = max(n_tiles, len);
        used[i] = false;
    }
    return n_tiles;
}
int domino_rec(int last){
    int n_tiles = 0, len;
    for(int i = 0; i < t.size(); i++){
        if(!used[i] && last == t[i].a){
            used[i] = true;
            len = 1 + domino_rec(t[i].b);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
        if(!used[i] && last == t[i].b){
            used[i] = true;
            len = 1 + domino_rec(t[i].a);
            n_tiles = max(n_tiles, len);
            used[i] = false;
        }
    }
    return n_tiles;
}
```

```
vector<tile> t    = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
vector<bool> used = {  false, false,  false,  false};
```

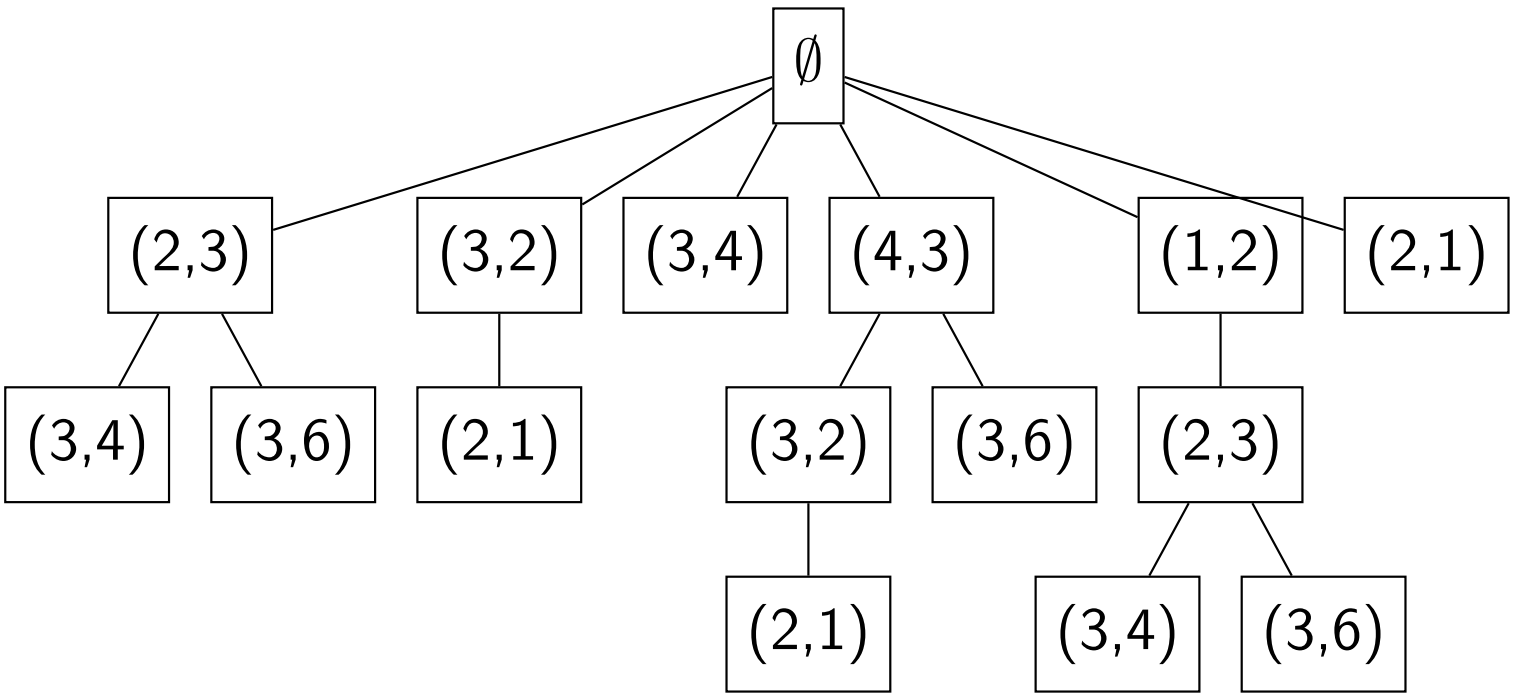
```
domino()
    n_tiles = 5;
    i = 4;
```

Cosa succede?

Idea

- Stiamo esplorando l'albero delle possibili soluzioni
- Quando una soluzione non è ammissibile ci fermiamo e risaliamo

```
vector<tile> t = {{2, 3}, {3, 4}, {1, 2}, {6, 3}};
```



Domande?

Esercizi

- Day 2