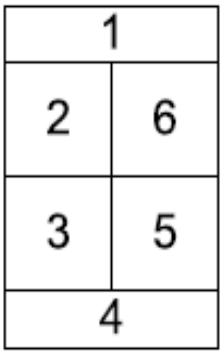
TECNICA DEL BACKTRACKING

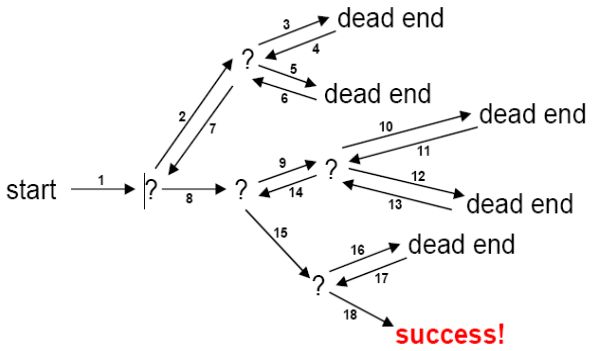
La tecnica di *backtracking* estende la ricerca esaustiva nella risoluzione di problemi di ricerca attraverso l'introduzione di alcuni controlli per verificare il più presto possibile se una soluzione in via di costruzione soddisfi o meno le condizioni di ammissibilità in modo da ridurre lo spazio di ricerca. Il problema tipico di ricerca risolto con il backtracking consiste nell'assegnare un valore, preso da un dominio *V*, a ciascuno degli *n* elementi di un vettore soluzione *X* in modo che siano soddisfatti alcuni vincoli su tali valori. Mentre la ricerca esaustiva ad ogni passo genera una soluzione possibile effettuando un possibile assegnamento di valori a tutti gli elementi di *X* e poi verifica i vincoli per essa, il backtracking assegna un valore ad un elemento la volta e verifica i vincoli immediatamente senza aspettare di aver assegnato un valore a tutti gli elementi. In questo modo può essere ridotto enormemente il numero di soluzioni ammissibili generate.

Esempio tipico: Colorazione di una mappa.
Si supponga di avere a disposizione una mappa geografica con la definizione dei confini tra i vari paesi, si vogliono colorare i vari paesi considerando i seguenti requisiti:

- Si vogliono usare al più tre colori: Rosso, Verde, Blu
- Paesi confinanti devo avere colori diversi
- La scelta di un colore per ciascun paese è puramente arbitraria
- La scelta di un colore per un paese, ha implicazioni sulla scelta dei possibili colori per gli altri paesi
- Possono esserci più colorazioni valide



Il problema della colorabilità di una mappa è un problema tipico. Molti altri problemi possono essere ricondotti a questo. Si noti che l'esistenza o meno di una soluzione dipende sia dal numero di colori a disposizione, sia dalla conformazione dei confini.
Un tipico processo di ricerca della soluzione con la tecnica del backtracking è il seguente:

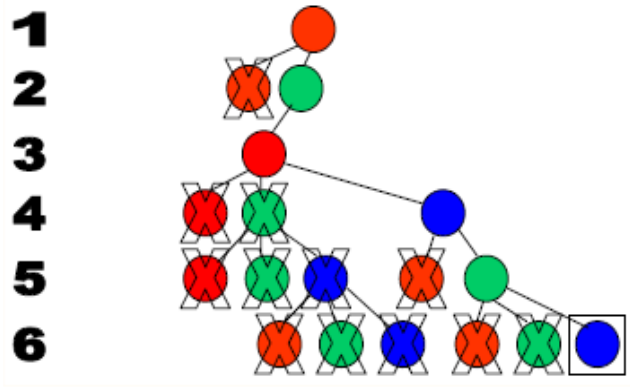


A partire dalla soluzione vuota, si procede con scelte parziali successive (nell'esempio, si inizia a colorare il paese 1 con uno dei colori disponibili) e si verifica se si può analogamente assegnare un valore di soluzione per i sottoproblemi rimanenti (la colorazione dei paesi 2,3,4, 5 e 6). Se ad un certo punto ci si accorge che le scelte fatte in passato non portano a nessuna soluzione ammissibile, si torna indietro un passo alla volta e si cercano soluzioni alternative.

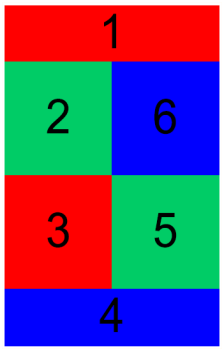
REQUISITI.
Per poter applicare la tecnica del Backtracking ad un problema devono valere le seguenti condizioni:

- Lo spazio delle soluzioni deve essere finito (l'insieme delle possibili configurazioni da verificare deve essere finito)
- La soluzione può essere rappresentata tramite una sequenza di soluzioni parziali s1, s2, ..., sn
- La dimensione (massima) della soluzione – n – è nota a priori
- Il valore di ciascuna sotto-soluzione s_i deve appartenere ad un insieme finito di valori

Torniamo al problema della colorazione della mappa introdotta precedentemente. Un possibile percorso di scelte successive potrebbe essere il seguente:



Notiamo subito che la prima scelta di assegnare il colore rosso al paese 1 va bene, mentre la prima scelta di assegnare il colore rosso al paese 2 va già in conflitto con i vincoli del problema, quindi bisogna fare un passo indietro e cercare un nuovo colore per il paese 2. Alla fine del processo, la mappa sarà così colorata:



In questo problema, ci sono 6 sotto-soluzioni; ciascuna corrisponde al colore da assegnare all'i-esimo paese. In ogni sotto-soluzione abbiamo uno spazio finito di valori ammissibili: rosso, blu, verde.
Nella tecnica del backtracking, si assegna un *ordine* (puramente arbitrario in molti casi, rilevante ai fini del problema in altri) tra i possibili valori di ciascuna soluzione, in modo da poter ottenere una sistematizzazione della tecnica.

Di seguito vediamo due possibili formulazioni (in pseudo codice) dell'algoritmo di backtracking: una ricorsiva ed una iterativa:

VERSIONE ITERATIVA

```
List sol = {};
boolean solve(sol) {
        x = MIN_VAL;
        boolean stop=false, existsSolution;
        while (!stop)
                if (x <= MAX_VAL)
                        if (canAdd(x,sol)) {
                                add(x,sol); // aggiungi x in coda a sol
                                if (isComplete(sol)) {
                                        existsSolution = true; stop= true;
                                }
                                else x = MIN_VAL;
                        }
                        else x = next(x);
                else {
                        if (isEmpty(sol)){
                                        existsSolution = false; stop= true;
                          }
                        else {
                                remove(x,sol);
                                x = next(x); //backtrack
                                }
                        }
        return existsSolution;
}
```

VERSIONE RICORSIVA

```
List sol = {};
boolean solve(sol) {
        x = MIN_VAL;
        while (x <= MAX_VAL)
                if (canAdd(x,sol)) {
                        add(x,sol); //aggiungi x in coda a sol
                        if (isComplete(sol)) return true;
                        elseif (solve(sol)) return true;
                        remove(x,sol); x = next(x); //backtrack
                }
                else
                        x = next(x);
        return false;
}
```

In questi due schemi, basta implementare le funzioni canAdd, add, remove, next e isComplete per come richiesto dalla formulazione del problema per poter realizzare una tecnica di backtracking.

Di seguito presentiamo una soluzione del problema della tre-colorabilità introdotto precedentemente.

Input: Si suppone che la mappa sia rappresentata da una matrice di booleani G in cui ciascuna riga/colonna rappresenta un paese, e G[i,j]=true indica che i paesi i e j sono confinanti.

Abbiamo già individuato precedentemente che lo spazio delle soluzioni candidate è composto da:

- Rosso --> MIN_VAL
- Verde
- Blu --> MAX_VAL

Chiaramente questo insieme si può rappresentare con un tipo enumerativo:

enum colors {Rosso=1, Verde, Blu};

e quindi MIN_VAL=1 e MAX_VAL=3

La soluzione *sol* è quindi un elenco di numeri (es. sol={1,2,1,3,2,3}) in cui sol[i] rappresenta il colore del paese *i*.

Vediamo di seguito le implementazioni (sempre in pseudocodice) delle funzioni mancanti:

Condizione di chiusura

```
bool isComplete(List sol) {
return size(sol) == size(G);
}
```

operatore next

```
int next(x) { return x+1; }
```

Verifica della correttezza dell'ultimo assegnamento

```
bool canAdd(x,sol) {
int size = size(sol);  // Sto verificando se posso inserire la
                       // soluzione x nella posizione size+1
     for (int i = 1 ; i <= size; i++)
          if (G[i, size+1] && sol[i] == x)
          //Se esiste un confine tra i e size+1 ed il colore di i è
          //già x, non posso assegnare x a size+1
               return false;
     return true;
}
```

La complessità della funzione di backtracking per il problema della colorazione di una mappa ha complessità $\Theta(n\,k^n)$, dove n è il numero di nodi nel grafo e k è il numero di colori disponibili.