Backtracking

- Tecnica di risoluzione esaustiva per problemi di ricerca.
- Introduce dei **controlli**:
  - Servono a scartare *tempestivamente* una soluzione (in via di costruzione) che non soddisferà i criteri di ammissibilità.
- Differenza con una tecnica esaustiva classica: **problema 3 colorabilità**
  - Soluzione esaustiva: *(1) assegno tutti i possibili colori a tutti i possibili nodi*, *(2) controllo ogni assegnamento*
  - Soluzione backtracking: *(1) costruisco una soluzione dando un colore ad un nodo*, *(2) se nella costruzione due nodi adiacenti hanno lo stesso colore → la soluzione viene scartata*
- Il backtracking è un algoritmo generale che **costruisce in modo incrementale le soluzioni** e **scarta quelle non valide**.
  - Generalmente usato quando il problema pone dei vincoli di soddisfacimento (*constraint satisfaction problems*)

Vademecum dello studente appassionato di backtracking

- Definizione della soluzione
  - Da cosa è rappresentata e cosa contiene (quali sono gli elementi che concorrono alla creazione della soluzione?)
- Dominio degli elementi della soluzione (`x_min` e `x_max`?)
- Criteri di `canAdd()` e `isComplete()`
  - `canAdd`: quando posso aggiungere un nuovo elemento alla soluzione?
  - `isComplete`: quando una soluzione creata è completa?
  - Nota: in molti problemi, la correttezza della soluzione si assicura nella `canAdd()`
- Funzioni `next()`, `add()` e `remove()`

Subset sum

- Dato un insieme di numeri interi positivi $\mathrm{S} = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\}$ determinare se esiste un sottoinsieme $\mathrm{R} \subseteq \mathrm{S}$ tale che la somma dei numeri contenuti in R sia esattamente uguale ad una costante data W.
- Esempio:
  - $\mathrm{S} = \{4, 12, 3, 88, 192\}$, $\mathrm{W} = 207$
  - Una possibile soluzione è $\mathrm{R} = \{12, 3, 192\}$

Subset sum + backtracking

```
1. bool solve(sol) {
2.   x = MIN_VAL;
3.   while (x <= MAX_VAL) {
4.     if (canAdd(x,sol)) {
5.       add(x,sol);
6.
7.       if (isComplete(sol))
8.         return true;
9.       elseif (solve(sol))
10.        return true;
11.
12.      remove(x,sol);
13.      x = next(x);
14.    } else
15.      x = next(x);
16.  }
17.  return false;
18. }
```

Soluzione:

- L'array sol rappresenta la soluzione R
- sol[i] = i è l'indice di un valore in S
- MIN_VAL = 0, MAX_VAL = S.len - 1 (i possibili valori contenuti in S)
- Nota: sol può essere grande al più S.len

canAdd & isComplete:

- canAdd: posso aggiungere il valore S[x] a sol sse
  - non è stato già inserito in sol, e
  - sum(sol) + S[x] ≤ W
- isComplete: completa quando sum(sol) == W

next & remove:

- next: prossimo indice (x+1)
- remove: rimuovo l'ultimo indice inserito

Subset sum

- Dato un insieme di numeri interi positivi $\mathrm{S} = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\}$ determinare se esiste un sottoinsieme $\mathrm{R} \subseteq \mathrm{S}$ tale che la somma dei numeri contenuti in R sia esattamente uguale ad una costante data W.
- Esempio:
  - $\mathrm{S} = \{4, 12, 3, 88, 192\}$, $\mathrm{W} = 207$
  - Una possibile soluzione è $\mathrm{R} = \{12, 3, 192\}$

- *Variante del problema*
- Dato un insieme di numeri interi positivi $\mathrm{S} = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\}$ determinare se esiste **il più piccolo** sottoinsieme $\mathrm{R} \subseteq \mathrm{S}$ tale che la somma dei numeri contenuti in R sia esattamente uguale ad una costante data W.
  - Come si risolve?

Appello Novembre 2018

- https://prototypes.mat.unical.it/asd/traccia_novembre2018.pdf
- Data una insieme di stringhe D, dire se esiste un sottoinsieme P di D grande esattamente q tale che tutte le stringhe in questo sottoinsieme siano congiunte fra loro
- Template classico:
  - Dato un insieme X e un numero k, dire se esiste un sottoinsieme Y di X, di cardinalità k, tale che valga una certa proprietà per ogni elemento di Y.
- Backtracking
  - La soluzione è il sottoinsieme Y ed è grande esattamente k,
  - L'insieme X rappresenta il dominio della soluzione => gli elementi della soluzione appartengono ad X,
  - canAdd: aggiungo un elemento alla soluzione se esso non viola la proprietà che stiamo verificando,
  - isComplete: la soluzione è grande k? (poiché la verifica della proprietà la facciamo già nella canAdd!)