Backtracking

- Căutare exhaustivă în spațiul stărilor
- Mai eficient decât forța brută întrucât drumurile (soluțiile parțiale) care nu mai pot duce la soluție sunt abandonate (cu forța brută ele ar fi construite până la sfârșit și abia apoi testate)

Permutări - Exemplu de urmărit pe ocw

Exemplu:

n = 3

Permutări: 1 2 3, 1 3 2, 2 1 3, 2 3 1, 3 1 2, 3 2 1

- Fiecare bkt are propriile sale domain si solution: ineficient spațial
- Daca transmit prin referință aceleași domain și solution de la un apel la altul: atenție să le refac atunci când mă întorc în apelul părinte
- Pruning – pot câștiga mult în eficiență dacă tai din domeniu acele valori care nu mai pot fi alese

Algoritmul general de backtracking

```
back(step, solution, domain) // eventual și alte argumente
    condiție_de_oprire
    for (value : domain)
      // aici facem un eventual pruning
      solution.push(value)
      check
      back(step + 1, solution, domain)
      // dacă nu alegem acest value, atunci el trebuie readus în domeniu
      // și soluția revine la cum era înainte să adăugăm value
      refacere_stare_anterioară
```

Combinări

Exemplu:

n = 3, k = 2

Combinări: 1 2, 1 3, 2 3

Aranjamente

Exemplu:

n = 3, k = 2

Aranjamente: 1 2, 1 3, 2 1, 2 3, 3 1, 3 2

Submulțimi

Exemplu:

n = 3

Submulțimi: {}, 1, 1 2, 1 2 3, 1 3, 2, 2 3, 3

Așezarea reginelor pe tabla de șah

Exemplu:

n = 5

O soluție: 1 5, 2 2, 3 4, 4 1, 5 3 sol[i] = coloana pe care se află regina de la linia i

X				
		X		
				X
	X			
			X	

Generare de șiruri

caractere = string de lungime n

freq = vector de n frecvențe (pentru fiecare caracter din caractere)

k = număr maxim acceptat de apariții consecutive ale unui același caracter

Care sunt toate șirurile care se pot genera folosind aceste caractere cu aceste frecvențe (respectând restricția că nu pot fi mai mult de k apariții consecutive ale aceluiași caracter)?

Exemplu:

caractere = „ab”

freq = {2, 3}

k = 2

Soluții: ababb, abbab, baabb, babab, babba, bbaab, bbaba

{} -> {a} -> {aa} -> {aab} -> {aabb} -> {aabbb-NUUU}

- ➔ {ab} -> {aba} -> {abab} -> {ababb}
- ➔ -> {abb} ->