FIȘĂ DE LUCRU · CLASA A XI-A · FIȘA 4 · ACOPERĂ LECȚIILE XI.6–XI.8

Backtracking: permutări, aranjamente, combinări, submulțimi, partiții

Durată: 110 min Nivel: mate-info

CE VEI ȘTI DUPĂ LECȚIE

- Recunoști generarea din enunț și numeri soluțiile cu formulă sau pe blocuri.
- Răspunzi la grilele BAC: a câta soluție, ce urmează după, ce este înainte, câte au o proprietate.
- Adaptezi schema `back(k)` la condiții noi: sumă dată, vecini, poziții interzise.

Ce trebuie să știi (o pagină)

Generare (în ordine lexicografică)	Condiția pe `x[k]`	Câte soluții	Exemplu
produs cartezian, K poziții, n valori	orice valoare	n^K	3 valori, 4 poziții: 81
permutări	distinct de `x[1..k−1]`	n!	n = 5: 120
aranjamente A(n, K)	distinct, doar K poziții	n!/(n − K)!	A(5, 3) = 60
combinări C(n, K)	`x[k] > x[k−1]`	n!/(K!(n − K)!)	C(7, 4) = 35
submulțimi (nevide)	`x[k] > x[k−1]`, afișezi la **fiecare** pas	$2^n - 1$	n = 5: 31
partițiile lui n	`x[k] ≥ x[k−1]`, suma = n	(se numără pe arbore)	n = 7: 15
plata sumei S	termeni nedescrescători din bancnotele date, suma = S	(se numără pe arbore)	S = 4 cu 1, 2: 3 moduri

Întrebare BAC	Metoda
„a câta soluție este ...?”	**blocuri**: pe poziția 1, fiecare valoare mai mică „acoperă” toate soluțiile restului; aduni blocurile sărite, apoi treci la poziția 2
„care este soluția a k-a?”	invers: împarți k − 1 la mărimea blocului (câtul = a câta valoare rămasă, restul = merge mai departe)
succesorul unei permutări	de la dreapta, sari peste sufixul descrescător; mărești elementul de dinainte cu cel mai mic mai mare din sufix; sufixul devine crescător

Întrebare BAC	Metoda
succesorul unei combinări	cea mai din dreapta poziție i cu `x[i] < n - K + i`; o mărești cu 1, iar după ea pui valori consecutive

> **SFAT**
>
> Înainte de orice calcul, scrie primele 3–4 soluții și verifică că ele coincid cu cele din enunț: așa afli **tipul** generării și **ordinea**.

Nivelul 1 · Prezice

★ citește și prezice

P1. Se generează în ordine alfabetică toate anagramele cuvântului `amor` (primele: `amor`, `amro`, `aomr`). Care este a 15-a? Câte încep cu `o`? A câta este `mora`? A câta este `roma` și ce cuvânt se generează imediat înaintea lui?

P2. Se generează toate podiumurile (aur, argint, bronz) cu sportivii `A, B, C, D, E`, în ordine alfabetică: `ABC`, `ABD`, `ABE`, `ACB`, … Câte sunt? Care este al 37-lea? Ce podium urmează după `BED`? Al câtelea este `DAC`? În câte podiumuri apare `A`?

P3. Se generează echipele de câte 4 elevi dintr-o grupă de 7 (elevii 1..7), ca la combinări. Ce echipă urmează după `1 5 6 7`? A câta este `2 3 4 5`? Care este ultima echipă care îl conține pe 3 ca cel mai mic elev? Câte echipe îi conțin și pe 1, și pe 7?

P4. Se generează, în ordine crescătoare, toate numerele de 4 cifre formate doar cu cifrele `0, 2, 5` (primele: 2000, 2002, 2005, 2020). Câte sunt? Care este al 20-lea? Câte sunt divizibile cu 5? Ce număr se generează imediat înainte de 5000?

P5. Se generează submulțimile nevide ale lui {1, 2, 3, 4, 5} ca în lecție (`1`, `1 2`, `1 2 3`, `1 2 3 4`, `1 2 3 4 5`, `1 2 3 5`, …). Câte sunt? A câta este `{2, 4}`? Ce urmează după `{1, 2, 5}`? Dar după `{2, 4, 5}`? Scrie submulțimile cu suma 8, în ordinea în care apar.

P6. Partițiile lui 7, generate ca în lecție (`1 1 1 1 1 1 1`, `1 1 1 1 1 2`, `1 1 1 1 3`, …). Câte sunt? A câta este `1 2 4`? Ce urmează după ea? Câte partiții au exact trei termeni?

P7. Ce afișează programul? Ce generează, în cuvinte?

```
int x[20], n = 4, nr;
void back(int k) {
    for (int v = 0; v <= 1; v++) {
        x[k] = v;
        if (k > 1 && x[k - 1] == 1 && x[k] == 1) continue;
        if (k == n) {
            nr++;
            if (nr == 5) { for (int i = 1; i <= n; i++) cout << x[i]; cout << ' '; }
        }
        else back(k + 1);
    }
}
int main() { back(1); cout << nr; }
```

P8. Pentru permutările lui {1, 2, 3} generate cu funcția `valid` (fără vector „folosit”), de câte ori se execută `x[k] = v` și de câte ori `valid` întoarce `false`?

```
void back(int k) {
    for (int v = 1; v <= n; v++) {
        x[k] = v;
        if (valid(k)) { if (k < n) back(k + 1); else afis(); }
    }
}
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Elevul vrea combinările de 3 luate câte 2 (`12 13 23`) și a scris `for (int v = x[k - 1]; v <= n; v++)` . Ce afișează programul? Ce generează, de fapt (dacă ignori valorile 0)? Repară.

P10. Ce afișează programul pentru n = 3? Ce rând lipsește?

```
void back(int k) {
    for (int v = 1; v <= n; v++)
        if (!folosit[v]) {
            x[k] = v; folosit[v] = 1;
            if (k == n) afis();
            else back(k + 1);
        }
}
```

P11. *Puzzle cu rânduri amestecate.* Funcția numără în câte moduri se poate plăti suma `S` cu bancnotele `b[1..m]` (oricâte din fiecare, ordinea nu contează). Apelul este `back(0, 1)` . Pune rândurile în ordine; un rând este în plus. Cât este `nr` pentru `b = (1, 2, 5)` , `S = 10` ? Ce ar număra funcția dacă ai folosi rândul în plus în locul celui corect?

```
A)  void back(int s, int p) {
B)          if (s + b[i] == S) nr++;
C)      for (int i = p; i <= m; i++)
D)          else back(s + b[i], i + 1);
E)  }
F)          if (s + b[i] <= S) {
G)          else back(s + b[i], i);
H)          }
```

P12. Completează golurile ca programul să genereze permutările în care **niciun** element nu stă pe poziția lui (`x[k] ≠ k`). Câte sunt pentru n = 4? Care este prima și care este ultima?

```
bool valid(int k) {
    if (__________) return false;
    for (int i = 1; i < k; i++) if (__________) return false;
    return true;
}
```

P13. *Ce greșeală a făcut elevul?* Corectează fiecare răspuns.

a. „Numerele de 3 cifre distincte formate cu cifrele 1, 3, 5, 7, 9 sunt C(5, 3) = 10.”

b. „La combinări de 5 luate câte 3, după `1 3 5` urmează `1 3 6` .”

c. „La submulțimile lui {1, 2, 3, 4} (ordinea din lecție), după `{1, 2, 3, 4}` urmează `{1, 3}` .”

P14. Programul trebuie să afișeze **a doua** soluție dintre aranjamentele de 4 luate câte 2 (adică `1 3`). Ce afișează de fapt? De ce? Repară mutând un singur rând.

```
int tinta = 2;
void back(int k) {
    int nr = 0;
    for (int v = 1; v <= n; v++) {
        x[k] = v;
        if (valid(k)) {
            if (k == K) { nr++; if (nr == tinta) afis(); }
            else back(k + 1);
        }
    }
}
```

Nivelul 3 · Scrie cod

★★ scrie cod

Fiecare program se scrie cu `back(k)`. Verifică **toate** testele.

P15. A câta permutare? Se dă o permutare a lui {1, ..., n} (n ≤ 8). Afișează a câta este în ordine lexicografică. Bonus: fără să generezi permutările (blocuri de (n − i)!).

`3` / `1 2 3`	`4` / `4 3 2 1`	`5` / `3 1 4 5 2`	`4` / `3 4 1 2`
`1`	`24`	`52`	`17`

P16. Echipe echilibrate. `n` elevi (n par, n ≤ 20) au punctajele date. Împarte-i în două echipe de câte n/2 astfel încât diferența dintre sumele punctajelor să fie minimă. Afișează diferența. Teste: `4` / `1 2 3 4` → `0`; `6` / `5 5 5 5 5 30` → `25`; `2` / `3 8` → `5`; `6` / `1 2 3 4 5 6` → `1`.

P17. Meniul cu buget. Un meniu are `m` feluri (m ≤ 6); pentru felul k se dau `c[k]` și cele `c[k]` prețuri. Alegi **câte un** preparat din fiecare fel. Câte meniuri costă cel mult `B` lei? (Taie ramura când suma depășește B.)

intrare	ieșire
`3` / `2 10 12` / `3 20 25 30` / `2 8 15` / `45`	`5`
același meniu, `B = 30`	`0`
`1` / `3 5 7 9` / `7`	`2`

P18. Parole fără litere dublate. Afișează, în ordine alfabetică, toate parolele de lungime `n` cu literele `a`, `b`, `c`, fără două litere vecine egale, apoi numărul lor. Teste: `1` → `a b c` (câte una pe rând) și `3`; `2` → `ab ac ba bc ca cb` și `6`; `3` → 12 parole, prima `aba`, ultima `cbc`.

P19. Turnul de cuburi. Se dau `n` cuburi (n ≤ 20) cu înălțimile date și o înălțime `H`. În câte moduri poți alege cuburi (fiecare cel mult o dată) care, puse unul peste altul, au exact înălțimea H? Teste: `5` / `2 3 5 7 8` / `10` → `3`; aceleași cuburi, `H = 1` → `0`; `3` / `1 1 1` / `2` → `3` (cuburile sunt diferite, chiar dacă au aceeași înălțime).

P20. Ce urmează după? Se dau `n`, `K` și o combinare `x[1..K]`. Afișează combinarea care urmează în ordine lexicografică, sau `ultima`, **fără** să le generezi pe toate. Teste: `5 3` / `1 4 5` → `2 3 4`; `5 3` / `3 4 5` → `ultima`; `6 2` / `2 3` → `2 4`; `4 4` / `1 2 3 4` → `ultima`.

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. A suta permutare. Care este a 100-a permutare a lui {1, 2, 3, 4, 5} în ordine lexicografică? Găsește-o cu blocuri (24, 6, 2, 1), fără să scrii nicio listă, apoi verifică cu un program.

P22. Câte, fără să generezi. Câte permutări ale lui {1, ..., 5} **nu** au 1 pe prima poziție **și nici** 5 pe ultima? (Atenție: dacă scazi ambele cazuri, pe unele le-ai scăzut de două ori.) Câte permutări ale lui {1, ..., 6} au 1 și 2 vecine? În câte apare 1 înaintea lui 2?

P23. Cât câștigi cu tăierea? La problema celor 8 regine, varianta „naivă" pune câte o regină pe fiecare linie, oriunde, și verifică abia la final. Câte configurații complete verifică? Varianta din lecție verifică la fiecare pas. Modifică programul ca să numere de câte ori se execută `x[k] = v`. Ce obții și de câte ori e mai rapid?

P24. Problemă inversă. Se generează toate numerele de 3 cifre **distincte**, cu cifre dintr-o mulțime necunoscută de cifre nenule, în ordine crescătoare. Primele patru sunt `235`, `237`, `238`, `253`. Care este mulțimea? (Justifică de ce nu poate conține și alte cifre.) Câte numere se generează? Care este al 30-lea și care este ultimul?

P25. Cântarul cu trei greutăți. Ai greutățile 1, 3 și 9 kg. Fiecare poate sta pe talerul cu marfa (−1), pe celălalt taler (+1) sau deloc (0), deci ai un produs cartezian cu valorile −1, 0, +1. Scrie programul care afișează, pentru fiecare variantă cu rezultat pozitiv, masa cântărită. Ce mase poți cântări și în câte moduri pe fiecare? Ce observi? Ce greutate ai adăuga ca să cântărești tot până la 40 kg?

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] `#123` **permutari** — ușoară
  https://www.pbinfo.ro/probleme/123/permutari
- [ ] `#196` **aranjamente** — ușoară
  https://www.pbinfo.ro/probleme/196/aranjamente
- [ ] `#197` **combinari** — ușoară
  https://www.pbinfo.ro/probleme/197/combinari
- [ ] `#320` **partitiinumar** — ușoară
  https://www.pbinfo.ro/probleme/320/partitiinumar

Temă pentru acasă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] `#3156` **perm-min-max** — ușoară
  https://www.pbinfo.ro/probleme/3156/perm-min-max
- [ ] `#1286` **submultimi1** — ușoară
  https://www.pbinfo.ro/probleme/1286/submultimi1

- [ ] #2245 **plata1** — ușoară
  https://www.pbinfo.ro/probleme/2245/plata1
- [ ] #344 **paranteze** — ușoară
  https://www.pbinfo.ro/probleme/344/paranteze
- [ ] #1810 **cifregen5** — ușoară
  https://www.pbinfo.ro/probleme/1810/cifregen5
