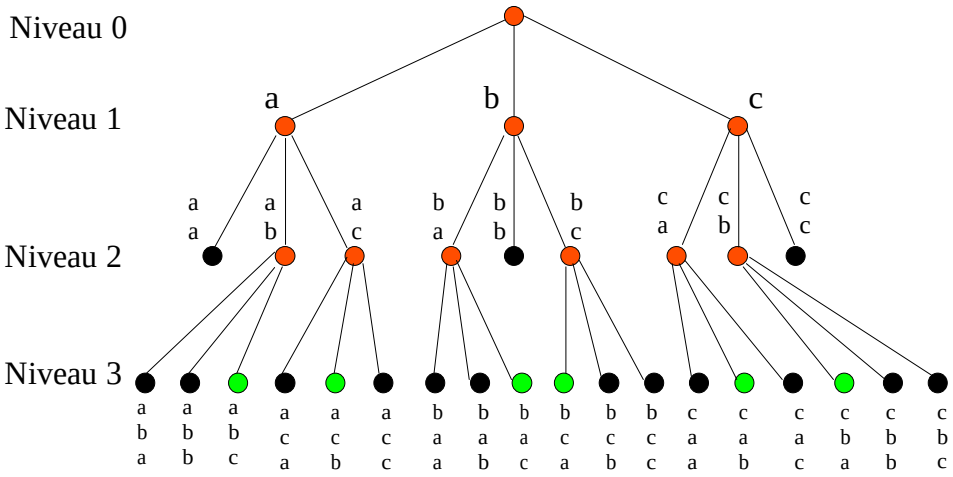
Permutations

Énoncé du problème

Il s'agit de concevoir le programme permettant d'obtenir toutes les permutations des *n* éléments d'un ensemble.

Ces solutions s'obtiennent en construisant un arbre dans lequel les feuilles porteront des permutations complètes et les nœuds non terminaux, des solutions partielles.

Ex : Soit l'ensemble *E = (a, b, c)*, l'arbre de recherche est le suivant :



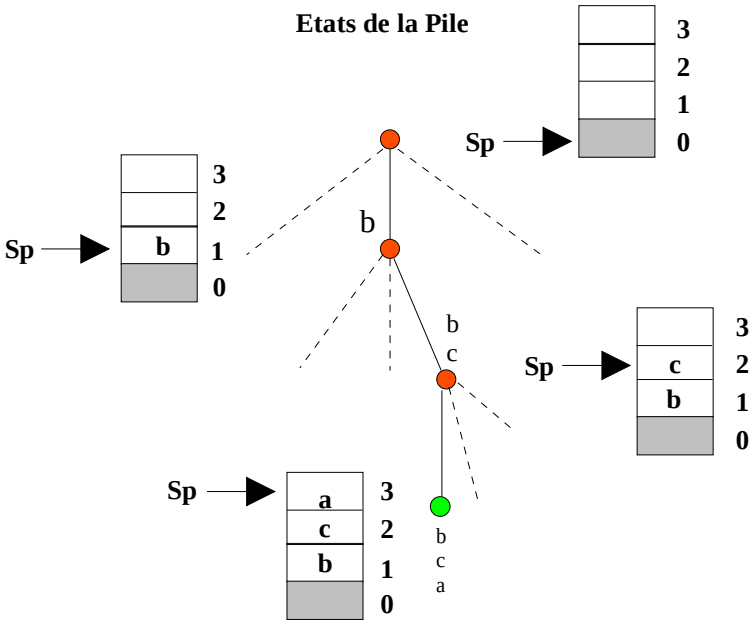
Lorsqu'on construit cet arbre, on peut directement éliminer les nœuds portant deux éléments identiques. Ainsi dans l'exemple ci-dessus, le nœud « *aa* » n'est pas valide : l'analyse de la branche qui y arrive peut être abandonnée.

Énumération partielle d'un arbre de décision. Algorithme de principe

La recherche des permutations des éléments d'un ensemble est un problème d'énumération partielle d'un arbre de décision.

La recherche de toutes les solutions s'effectue par un parcours en profondeur d'abord (*Depth First Search*). L'utilisation d'une pile est alors nécessaire afin de conserver la trace des choix déjà effectués (à partir de chaque nœud, il y a *n* choix possibles, à priori).

Exemple



Pour le problème des permutations, un nœud sera valide lorsque la liste des choix effectués ne comportera pas deux éléments identiques, ou encore lorsque le choix qui vient d'être fait, n'avait pas encore été fait.

Algorithme

```
Le parcours de l'arbre commence au niveau 0 (la pile est vide).

Le processus de recherche des solutions est amorcé par la prise de la
première décision: on passe au niveau 1 de l'arbre en choisissant le premier
élément de l'ensemble E.

Répéter

     1)  Tant que le nœud courant est valide Faire
            Si c'est un nœud terminal Alors
               _ enregistrer ou imprimer la solution
               _ Sortir de 1) Tant que pour aller au 2) Tant que

            Sinon /* pas un nœud terminal */
               Passer au premier fils de ce nœud (passer au niveau suivant
               en choisissant le premier élément de l'ensemble E)
            Fin Si
         Fin Tant que

     2)  Tant que la recherche n'est pas terminée et le nœud courant n'a
         plus de frère Faire

            Remonter au nœud père
         Fin Tant que

     3)  Si la recherche n'est pas terminée Alors
            Passer au nœud frère suivant
         Fin Si

Jusqu'à ce que la recherche soit terminée
```

Note

Soit « ***ab*** » le nœud courant, alors :

- « ***aba*** » est son premier nœud fils
- « ***a*** » est son nœud père
- « ***ac*** » est son nœud frère suivant

Travail à effectuer

Les éléments à permuter sont les caractères d'un mot fourni par l'utilisateur lors de l'exécution du programme. Dans le programme ces caractères seront désignés par leur **rang** dans le mot fourni par l'utilisateur.

Autrement dit, le programme ne manipulera pas directement les caractères mais leurs indices (des entiers) dans le processus de recherche des permutations. Ce n'est qu'au moment de l'affichage des solutions qu'on exploitera les caractères du mot.

On se limitera à des mots de 12 caractères au plus (*12!* solutions)

Le programme affichera toutes les solutions (*n* solutions par ligne, avec *n* dépendant de la longueur du mot), ainsi que le nombre de solutions obtenues.

Indications

Définir le type de la pile utilisée comme une structure.

Exemple

```
typedef  struct
{
     int    Sp;                /* Pointeur de pile */
     int    P[ MAXPILE + 1];   /* !!! Pile d'entiers   */
} T_PILE;
```

Définir sous forme de macro-fonctions les opérations suivantes :

- Passer au premier nœud fils
- Remonter au nœud père
- Passer au nœud frère suivant

Définir également sous forme de macro-fonctions les prédicats suivants :

- Le nœud courant est-il un nœud terminal ?

– Le nœud courant n'a t'il plus de nœud frère suivant ?

– La recherche n'est-elle pas terminée ?

Définir les fonctions dont les prototypes sont les suivants :

```
int Valide (T_PILE pile) ;
```

Cette fonction est chargée de vérifier que le choix qui vient d'être fait (sa valeur est présente au sommet de la pile) est valide ou non. Elle retourne 1 si le choix est valide et 0 sinon.

```
unsigned long Afficher (T_PILE pile, const char *mot) ;
```

Cette fonction est chargée d'afficher la solution qui vient d'être trouvée : la liste des choix effectués est rangée dans *pile*, et les caractères correspondants à afficher sont dans *mot*. Cette fonction retourne comme résultat le numéro de la solution affichée.

> **Indication :** Définir une variable statique dans la fonction Afficher**.**

Elle génère un saut de ligne lorsque le numéro de la solution affichée est un multiple du nombre de termes pouvant être affichés par ligne sur l'écran de la console d'affichage.

> **Note :** En mode console, on peut afficher jusqu'à 80 caractères par ligne.

Exemple

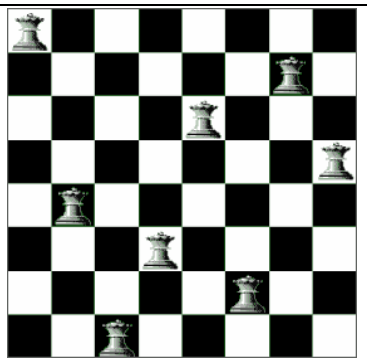
```
Entrez le mot à permuter: lacet

lacet lacte laect laetc latce latec lcaet lcate lceat lceta lctae lctea
leact leatc lecat lecta letac letca ltace ltaec ltcae ltcea lteac lteca
alcet alcte alect aletc altce altec aclet aclte acelt acetl actle actel
aelct aeltc aeclt aectl aetlc aetcl atlce atlec atcle atcel atelc atecl
claet clate cleat cleta cltae cltea calet calte caelt caetl catle catel
celat celta cealt ceatl cetla cetal ctlae ctlea ctale ctael ctela cteal
elact elatc elcat elcta eltac eltca ealct ealtc eaclt eactl eatlc eatcl
eclat eclta ecalt ecatl ectla ectal etlac etlca etalc etacl etcla etcal
tlace tlaec tlcae tlcea tleac tleca talce talec tacle tacel taelc taecl
tclae tclea tcale tcael tcela tceal telac telca tealc teacl tecla tecal

Durée de la recherche = 0.110 s

Il y a 120 solutions.
```

Les huit dames



Énoncé du problème

Il s'agit maintenant de concevoir et de réaliser le programme permettant d'obtenir les positions de 8 dames sur un échiquier en faisant en sorte qu'aucun couple de dames ne soit en « prise ».

Notes

On rappelle que dans le jeu d'échecs, une dame peut se déplacer en ligne droite sur toute ligne, colonne ou diagonale de l'échiquier. Deux dames sont donc « en prise » si elles sont situées sur la même ligne ou la même colonne ou encore la même diagonale.

Ce problème a été posé pour la première fois en 1848 par **Max Bezzel** dans un journal d'échecs de la *Berliner Schachgesellschaft*. Il a été résolu en 1850 par **Franz Nauck** et publié dans la revue allemande *Illustrierten Zeitung*. **Carl Friederich Gauss** (1777-1855), intéressé par cette « colle » n'aurait trouvé que 72 solutions.

Principe

En partant du principe que l'on ne peut mettre qu'une seule reine par ligne, le problème se ramène à la recherche des numéros de colonnes successifs (ou plus exactement des lettres de colonnes) utilisés pour placer les huit dames.

Indication

Le programme de recherche des permutations précédent (cf. page 328) permet le placement de n tours sur un échiquier $n \times n$. Une simple adaptation de ce programme permettra de traiter le problème des dames.

Travail à effectuer

Le programme à réaliser permettra à l'utilisateur de :

- fixer la taille de l'échiquier entre 4×4 et 15×15
- définir le mode de présentation des résultats, c'est-à-dire :
  - soit sous forme *a1b2c3*...
  - soit sous forme pseudo-graphique
  - soit sans affichage des solutions

Il fournira dans tous les cas le nombre de solutions trouvées, ainsi que la durée d'exécution de la phase de recherche.

Exemple 1

```
Entrez la taille de l'échiquier (15x15 max): 7
Choix de l'affichage
        Tapez 1 pour un affichage en notation échiquéenne
        Tapez 2 pour un affichage pseudo graphique
        Tapez 3 pour ne pas afficher les solutions
Votre choix : 1

a1c2e3g4b5d6f7 a1d2g3c4f5b6e7 a1e2b3f4c5g6d7 a1f2d3b4g5e6c7 b1d2a3g4e5c6f7
b1d2f3a4c5e6g7 b1e2a3d4g5c6f7 b1e2c3a4g5d6f7 b1e2g3d4a5c6f7 b1f2c3g4d5a6e7
b1g2e3c4a5f6d7 c1a2f3b4e5g6d7 c1a2f3d4b5g6e7 c1e2g3b4d5f6a7 c1f2b3e4a5d6g7
c1g2b3d4f5a6e7 c1g2d3a4e5b6f7 d1a2c3f4b5g6e7 d1a2e3b4f5c6g7 d1b2g3e4c5a6f7
d1f2a3c4e5g6b7 d1g2c3f4b5e6a7 d1g2e3b4f5a6c7 e1a2d3g4c5f6b7 e1a2f3d4b5g6c7
e1b2f3c4g5d6a7 e1c2a3f4d5b6g7 e1g2b3d4f5a6c7 e1g2b3f4c5a6d7 f1a2c3e4g5b6d7
f1b2e3a4d5g6c7 f1c2a3d4g5e6b7 f1c2e3g4a5d6b7 f1c2g3d4a5e6b7 f1d2b3g4e5c6a7
f1d2g3a4c5e6b7 g1b2d3f4a5c6e7 g1c2f3b4e5a6d7 g1d2a3e4b5f6c7 g1e2c3a4f5d6b7


Durée de la recherche =     0.000 s

Il y a 40 solutions
```

Exemple 2

```
Entrez la taille de l'échiquier (15x15 max): 4
Choix de l'affichage
        Tapez 1 pour un affichage en notation échiquéenne
        Tapez 2 pour un affichage pseudo graphique
        Tapez 3 pour ne pas afficher les solutions
Votre choix : 2

Solution 1

      A   B   C   D
    +---+---+---+---+
  1 | . | R | . | . |
    +---+---+---+---+
  2 | . | . | . | R |
    +---+---+---+---+
  3 | R | . | . | . |
    +---+---+---+---+
  4 | . | . | R | . |
    +---+---+---+---+

Solution 2

      A   B   C   D
    +---+---+---+---+
  1 | . | . | R | . |
    +---+---+---+---+
  2 | R | . | . | . |
    +---+---+---+---+
  3 | . | . | . | R |
    +---+---+---+---+
  4 | . | R | . | . |
    +---+---+---+---+


Durée de la recherche =     0.000 s

Il y a 2 solutions
```

Exemple 3

```
Entrez la taille de l'échiquier (15x15 max): 14
Choix de l'affichage
        Tapez 1 pour un affichage en notation échiquéenne
        Tapez 2 pour un affichage pseudo graphique
        Tapez 3 pour ne pas afficher les solutions
Votre choix : 3


Durée de la recherche =    38.725 s

Il y a 365596 solutions
```