Interrogation d'informatique n°1. Corrigé

Exercice

```
def somme(A,B) :
        n = A["dim"] ; C = {"dim":n}
        for x in A :
             if (x != "dim") and (x in B) :
             v = A[x] + B[x]
             if v != 0 :
                C[x] = A[x] + B[x]
        # il reste à traiter les cas où A[i,j] = 0 et B[i,j] ≠ 0
        for x in B :
             if not(x in A) : C[x] = B[x]
```

Problème

1)
```
def choice(L) :
       n = len(L)
       return L[random.randint(0,n)]
```

2)
```
def positions_possibles(p,atteints) :
       liste = [] ; (x,y) = p
       for q in [(x+1,y),(x-1,y),(x,y-1),(x,y+1)] :
            if not (q in atteints) :  liste.append(q)
       return liste
```

2) bis) On prend une courbe en escargot de longueur 7. Elle est entièrement déterminée par le choix du premier coude (p_0, p_1, p_2) : il y a 4 solutions pour p_1, puis 2 choix pour p_2, donc 8 possibilités.

3)
```
def genere_chemin_naif(n) :
      atteints = [(0,0)] ;  p = (0,0)
      for _ in range(n) :
           L = positions_possibles(p,atteints)
           if L : p = choice(L)     ### si L n'est pas vide
          else :  return None
      return L
```

4) La complexité de `positions_possibles(p,atteints)` est linéaire en la longueur de la liste `atteints`. D'où une complexité quadratique en $O(n^2)$, car $\sum_{k=1}^{n}(1+k) = O(n^2)$.

5) On renvoie pour chaque valeur de $n \in [\![1, 300]\!]$, la proportion de constructions d'un CAE de longueur n qui n'a pas abouti. On voit donc que rapidement ce procédé n'est pas très efficace même pour des valeurs relativement petites de n.

6)
```
def est_CAE(chemin) :
    dico = {}
    for p in chemins :
        if (p in chemins) :  return False
        else p[x] = 1
    return True
```

7)
```
def rot(p,q,a) :
    (x0,y0) = p ; (x1,y1) = q ;
    if a == 0 :  return ( 2*x0 - x1 , 2*y0 - y1)
    if a == 1 :  return ( x0 - (y1-y0) , y0 + (x0-x1))
    if a == 2 :  return ( x0 + (y1-y0) , y0 - (x0-x1))
```

8)
```
def rotation(chemin,i_piv,a) :
    new_chemin = chemin[0 :  i_piv]
    p = chemin[i_piv]
    for q in chemin[1+i_piv :  ]  :
        new_chemin.append(rot(p,q,a))
    return new_chemin
```

9)
```
def genere_chemin_pivot(n,n_rot) :
    chemin = [(k,0) for k in range(n)] ; compteur = 0
    while compteur < n_rot :
        a = random.randint(0,3) ; i_piv = random.randint(1,n)
        new_chemin = rotation(chemin,i_piv,a)
        if est_CAE(chemin) :
            chemin = new_chemin
            compteur += 1
    return chemin
```