6 Dichotomie

July 26, 2024

1 Recherche dichotomique

[1]:
```
def dicho_search(el:int, li:[int])->int:
    """
    Retourne l'indice de el dans li.
    Retourne None si el n'est pas présent dans li.
    """
    imin = 0
    imax = len(li)-1
    while imax - imin >= 0:
        im = (imin + imax)//2
        if el == li[im]: # on a trouvé
            return im
        elif el > li[im]: # donc iel > im = imin
            imin = im + 1
        else:
            imax = im - 1
    return None # on a pas trouvé
```

[2]:
```
li = list(range(5))
li
```

[2]:
```
[0, 1, 2, 3, 4]
```

[3]:
```
for el in li:
    print(dicho_search(el, li))
```

```
0
1
2
3
4
```

2 Exponentiation rapide

[4]:
```
def exp(a:int, n:int)->int:
    """
    Retourne a^n.
    """
    P = 1
    for i in range(n):
        P *= a
    return P
```

[5]:
```
exp(2, 4)
```

[5]: 16

[6]:
```
def bin(n:int)->[int]:
    """
    Retourne la liste des bits de n dans l'ordre
    décroissant des puissances de 2
    """
    li = []
    while n != 0:
        li.append(n%2)
        n = n//2
    return li
```

[14]:
```
bin(11)
```

[14]: [1, 1, 0, 1]

[8]:
```
def horner(coeff:[int], x=2)->int:
    """
    Retourne la valeur du polynôme défini par la liste de coefficients
    coeff en x.
    """
    S = 0 # c'est une somme
    p = 1 # puissances de x, c'est un produit
    for c in coeff:
        S = S + p*c
        p = p*x # la puissance augmente de 1
    return S
```

[9]:
```
horner(bin(11))
```

[9]: 11

[10]:
```
def exp(a:int, n:int)->int:
    """
    Retourne a^n par l'algorithme d'exponentiation rapide.
    """
    P = 1 # c'est un produit
    k = a # a^(2^n), vaut a quand n=0
    coeff = bin(n) # décomp binaire de n
    for c in coeff:
        P = P * k**c
        k = k**2 # on mutiplie la puissance par 2 donc a^(2^n)->a^(2n+1)
    return P
```

[11]:
```
exp(3, 7), 3**7
```

[11]: (2187, 2187)

[12]:
```
def exp(a:int, n:int)->int:
    """
    Version finale :
    - plus besoin de stocker la decomp binaire de n
    - le terme k**c est calculé par le if (seulement deux résultats possibles)
    - le reste de la div par 2 en base 2 c'est juste le dernier bit (très␣
 ↪rapide)
    - le quotient de la div par 2 consiste à aller voir le chiffre binaire␣
 ↪suivant
    """
    P = 1
    k = a
    while n != 0:
        if n%2 == 1: # quand n%2==0 on ne fait rien (P=P)
            P = P*k
        k = k*k # le carré se calcule comme un produit
        n = n//2
    return P
```

[13]:
```
exp(3, 7)
```

[13]: 2187

[ ]:
```

```