SWERC NoteBook

SaintGermainDesPrés : Mathilde Bonin, Eyal Cohen, Hugo Demaret

March 20, 2022

1 Configuration

1.1 C/C++

2 Parcours de graphes

2.1 Implémentation des graphes

2.1.1 C/C++

2.1.2 Python

2.2 DFS - Depth First Search

```python
#version iterative pour eviter la recursion limit de python
def dfs_iterative(graph,start,seen):
    seen[start] = True
    to_visit = [start]
    while to_visit:
        node = to_visit.pop()
        for neighbour in graph[node]:
            if not seen[neighbour]:
                seen[neighbour] = True
                to_visit.append(neighbour)
```

2.3 BFS - Breadth First Search

```python
from collections import deque
def bfs(graph, start=0):
    to_visit = deque()
    dist = [float('inf')] * len(graph)
    prec = [none] * len(graph)
    dist[start] = 0
    to_visit.appendleft(start)
    while to_visit: #evalue a faux si vide
        node = to_visit.pop()
        for neighbour in graph[node]:
            if dist[neighbour] == float('inf'):
                dist[neighbour] = dist[node] + 1
                prec[neighbour] = node
                to_visit.appendleft(neighbour)
    return dist, prec
```

```
point = [x,y]
```

3.1.2 Cross-product

```
def cross_product(p1, p2):
  return p1[0] * p2[1] - p2[0] * p1[1]
```

3.1.3 Direction

```
def left_turn(a,b,c):
  return (a[0]-c[0]) * (b[1]-c[1]) - (a[1]-c[1]) * (b[0]-c[0]) > 0
  # If floats are used, instead of 0 test if in [0-10E-7,0+10E-7]
```

3.2 Enveloppe convexe

Complexité : $\mathcal{O}(n \log(n))$

```
def andrew(S):
    S.sort()
    top = []
    bot = []
    for p in S:
        while len(top) >= 2 and not left_turn(p,top[-1],top[-2]):
            top.pop()
        top.append(p)
        while len(bot) >= 2 and not left_turn(bot[-2],bot[-1],p):
            bot.pop()
        bot.append(p)
    return bot[:-1] + top[:0:-1]
```

```python
def pgcd(a,b):
    return a if b == 0 else pgcd(b,a%b)
```

5.2 Coefficients de Bézout

```python
def bezout(a,b):
    if b == 0:
        return (1,0)
    else:
        u,v = bezout(b,a%b)
        return (v, u - (a//b) *v)
def inv(a,p):
    return bezout(a,p)[0]%p
```

5.3 Coefficients binomiaux

```python
def binom(n,k,p):
    prod = 1
    for i in range(k):
        prod = (prod * (n-i)) // (i+1) %p
    return prod
#Enlever le p et mod p pour sans modulo
```