Devoir surveillé n°2 – NSI
Correction

Exercice 1

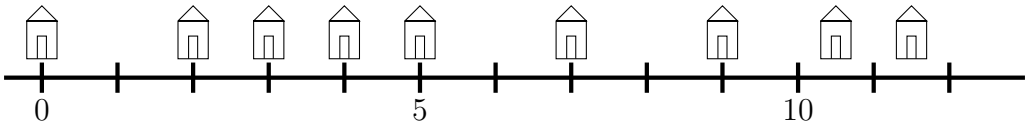
1. Les instructions sont :

```
m1 = Maison(1)
m2 = Maison(3.5)
```

2. Le code est :

```
a = Antenne(2.5, 1)
```

3. Le schéma est le suivant :



4. Le code complété :

```
def creation_rue(pos):
    pos.sort()
    maisons = []
    for p in pos:
        m = Maison(p)
        maisons.append(m)
    return maisons
```

5. Voici le code :

```
def couvre(self, maison):
    pos_ant = self.get_pos_antenne()
    pos_mai = maison.get_pos_maison()
    r = self.get_rayon()
    return abs(pos_ant - pos_mai) <= r
```

6. L'instruction

```
>>> maisons = creation_rue([0, 2, 3, 4, 5, 7, 9, 10.5, 11.5])
```

Crée la liste des maisons positionnées comme vu plus haut.
Ensuite,

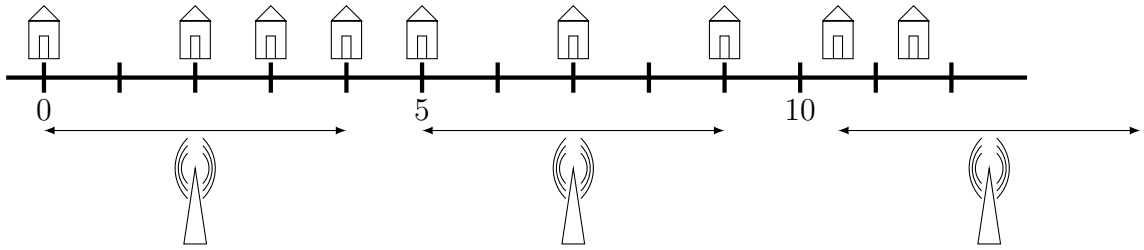
```
>>> antennes = strategie_1(maisons, 2)
```

Crée la liste des antennes construites à l'aide de la stratégie 1.
Enfin,

```
>>> print([a.get_pos_antenne() for a in antennes])
```

Affiche la liste des position des antennes obtenues précédemment, à savoir :
`[0, 3, 7, 10.5]`

7. Les maisons sont toujours à la même position.
   On trace les antennes obtenues par la stratégie 2 (avec leur rayon d'action pour justifier) :



   Les positions des antennes sont donc : `[2, 7, 12.5]`.

8. Le code de la fonction `strategie_2` :

```
def strategie_2(maisons, rayon):
  antennes = [Antenne(maisons[0].get_pos_maison()+rayon, rayon)]
  for m in maisons[1:]:
    if not antennes[-1].couvre(m):
      antennes.append(Antenne(m.get_pos_maison()+rayon, rayon))
  return antennes
```

   C'est le même, sauf qu'on place mieux les antennes, en décalant du rayon.

9. Comme les codes sont les mêmes en dehors du décalage (dont le coût est constant), leur coût est donc le même.
   Le coût est linéaire ($O(\mathtt{n})$) car on parcourt la liste des maisons de longueur `n`, et à chaque itération les instructions exécutées sont à coût constant.