corrigé

Sommes de sous-ensembles

1. Préliminaires

Question 1.

```
def V(n):
    v = [26]
    for i in range(n - 1):
        v.append((353 * v[-1]) % 997)
    return v
```

2. Stratégie optimale pour un jeu

Question 2.

```
def sommes(v):
    n = len(v)
    S = [[None for j in range(n)] for i in range(n)]
    for i in range(n):
        S[i][i] = v[i]
        for j in range(i + 1, n):
            S[i][j] = S[i][j - 1] + v[j]
    return S
```

Question 3. Si on choisit de prendre l'élément v_i, l'adversaire remporte $\mathrm{M}(i+1,j)$ points et nous

$$v_i + \mathrm{S}(i+1,j) - \mathrm{M}(i+1,j) = \mathrm{S}(i,j) - \mathrm{M}(i+1,j)$$

Si on choisit de prendre l'élément v_j, l'adversaire remporte $\mathrm{M}(i,j-1)$ points et nous

$$v_j + \mathrm{S}(i,j-1) - \mathrm{M}(i,j-1) = \mathrm{S}(i,j) - \mathrm{M}(i,j-1)$$

On en déduit que
$$\begin{aligned}\mathrm{M}(i,j) &= \max\big(\mathrm{S}(i,j) - \mathrm{M}(i+1,j), \mathrm{S}(i,j) - \mathrm{M}(i,j+1)\big)\\ &= \mathrm{S}(i,j) - \min\big(\mathrm{M}(i+1,j), \mathrm{M}(i,j-1)\big).\end{aligned}$$

Question 4. On calcule le tableau des $\mathrm{M}(i,j)$ en prenant garde à l'ordre des dépendances :

```
def gain(v):
    n = len(v)
    M = [[None for j in range(n)] for i in range(n)]
    S = sommes(v)
    for i in range(n):
        M[i][i] = v[i]
    for i in range(n - 1, -1, -1):
        for j in range(i + 1, n):
            M[i][j] = S[i][j] - min(M[i + 1][j], M[i][j - 1])
    return M[0][n - 1]
```

3. Sommes de sous-ensembles

Question 5. On définit tout d'abord une fonction qui génère la liste S_n :

```
def sousSommes(v, t):
    S = [0]
    for k in range(len(v)):
        T = []
        for x in S:
            if x + v[k] <= t:
                T.append(x + v[k])
        S = S + T
    return S
```

Il reste à calculer le nombre d'occurrences de t dans la liste S_n :

```
def nbDecompositions(v, t):
    S = sousSommes(v, t)
    nb = 0
    for x in S:
        if x == t:
            nb += 1
    return nb
```

Question 6.

```
def lenS(v, t):
    return len(sousSommes(v, t))
```

Pour tout $i \in [\![1, n]\!]$, $\operatorname{card} S_i \leqslant 2 \operatorname{card} S_i$ donc $\operatorname{card} S_i \leqslant 2^i$. Il y a égalité lorsque t est supérieur ou égal à la somme de tous les éléments de v, et dans ce cas $\operatorname{card} S_n = 2^n$. La complexité, tant temporelle que spatiale, est donc exponentielle.

Question 7.

```
def sousSomme2(v, t):
    S = {0: 1}
    for k in range(len(v)):
        T = {}
        for x in S:
            if x + v[k] <= t:
                if x + v[k] in T:
                    T[x + v[k]] += S[x]
                else:
                    T[x + v[k]] = S[x]
        for x in T:
            if x in S:
                S[x] += T[x]
            else:
                S[x] = T[x]
    return S
```

```
def nbDecompositions2(v, t):
    S = sousSomme2(v, t)
    if t in S:
        return S[t]
    else:
        return 0
```