

On crée cette fois, un dictionnaire pour stocker les informations.

```
1 prix = {1: 2, 2: 5, 3: 8, 4: 10, 5: 11, 6: 14, 8: 17, 10: 20}
```

On peut s'appuyer sur le problème du rendu de monnaie. L'approche naïve effectue un nombre d'appels conséquent.

```
1 def decoupe_naif(l: int, prix: dict) -> int:
2     """
3     renvoie le prix maximum pour une barre de taille l
4     """
5     if l == 0:
6         return 0
7     val_max = 0
8     for taille, valeur in prix.items():
9         if l >= taille:
10             val_max = max(val_max, decoupe_naif(l-taille, prix)+valeur)
11     return val_max
```

```
1 def decoupe_TD(l: int, prix: dict, track: int) -> int:
2     """
3     renvoie le prix maximum pour une barre de taille l
4     """
5     if track[l] > 0:
6         return track[l]
7     if l == 0:
8         track[0] = 0
9         return track[0]
10    val_max = 0
11    for taille, valeur in prix.items():
12        if l >= taille:
13            val_max = max(val_max, decoupe_TD(l-taille, prix, track)+
14                           valeur)
15    track[l] = val_max
16    return track[l]
17
18 track = [-1 for _ in range(longueur+1)]
```

```
1 def decoupe_BU(l: int, prix: dict) -> int:
2     track = [0 for _ in range(l+1)]
3     # calcule le prix pour chaque longueur en partant des petites valeurs
4     for i in range(1, l+1):
5         val_max = 0
6         for taille, valeur in prix.items():
7             if i >= taille:
8                 val_max = max(val_max, track[i-taille]+valeur)
9         track[i] = val_max
10    return track[l]
```