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

```
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.

```
def decoupe_naif(l: int, prix: dict) -> int:
    """
    renvoie le prix maximum pour une barre de taille l
    """
    if l == 0:
        return 0
    val_max = 0
    for taille, valeur in prix.items():
        if l >= taille:
            val_max = max(val_max, decoupe_naif(l-taille, prix)+valeur)
    return val_max
```

```
def decoupe_TD(l: int, prix: dict, track: int) -> int:
    """
    renvoie le prix maximum pour une barre de taille l
    """
    if track[l] > 0:
        return track[l]
    if l == 0:
        track[0] = 0
        return track[0]
    val_max = 0
    for taille, valeur in prix.items():
        if l >= taille:
            val_max = max(val_max, decoupe_TD(l-taille, prix, track)+
               valeur)
    track[l] = val_max
    return track[l]


track = [-1 for _ in range(longueur+1)]
```

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


Christophe Viroulaud