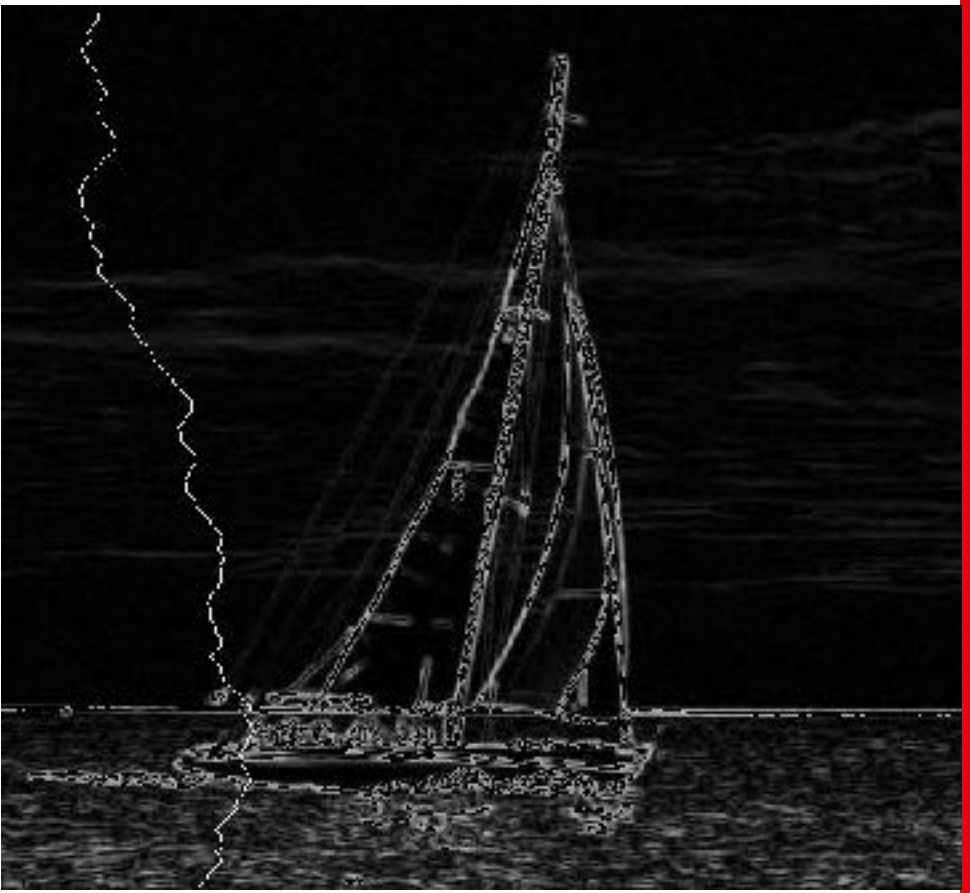


Information, Cacul et Communication

Partie Programmation

Cours 10 : Miniprojet – suite

24.11.2023

Patrick Wang

Au sommaire cette semaine

1. Rappel de la semaine dernière
2. Suite du miniprojet – Recherche de seam de moindre énergie

Au sommaire cette semaine

1. Rappel de la semaine dernière
2. Suite du miniprojet – Recherche de seam de moindre énergie

1. Rappel de la semaine dernière

- Création d'images avec `new_image_grey()` ou `new_image_rgb()`
- Découverte du filtre de lissage et du filtre de Sobel
- Travail de la séance d'exercices
  - `rgb_to_grey(r, g, b) -> int`
  - `to_grayscale(img) -> Image`
  - `clamp_index(index, length) -> int`
  - `apply_kernel(img_grey, kernel) -> Image`

1. Rappel de la semaine dernière


```python
if __name__ == "__main__":
    seam_carving("imgs/americascup.jpg", 200)
```

Objet de cette semaine

```python
def seam_carving(img_path: str, num_cols: int) -> None:
    name, ext = split_name_ext(img_path)
    folder = name + os.path.sep
    os.makedirs(folder, exist_ok=True)

    img = load_image(img_path)

    img_grey = to_grayscale(img)
    save_image(img_grey, folder + "grey" + ext)

    img_grey = smoothen(img_grey)
    save_image(img_grey, folder + "smooth" + ext)

    img_grey = sobel(img_grey)
    save_image(img_grey, folder + "sobel" + ext)
    img_grey = np.uint8(img_grey)

    for i in range(num_cols):
        seam = find_seam(img_grey)

        img_highlight = highlight_seam(img, seam)
        save_image(img_highlight, folder + f"highlight_{i}" + ext)
        img_grey_highlight = highlight_seam(img_grey, seam)
        save_image(img_grey_highlight, folder + f"highlight_{i}_grey" + ext)

        img = remove_seam(img, seam)
        save_image(img, folder + f"step_{i}" + ext)
        img_grey = remove_seam(img_grey, seam)
```

Au sommaire cette semaine

1. Rappel de la semaine dernière
2. Suite du miniprojet – Recherche de seam de moindre énergie

2. Recherche du seam de moindre énergie

Suite au lissage et filtre de Sobel :

2. Recherche du seam de moindre énergie

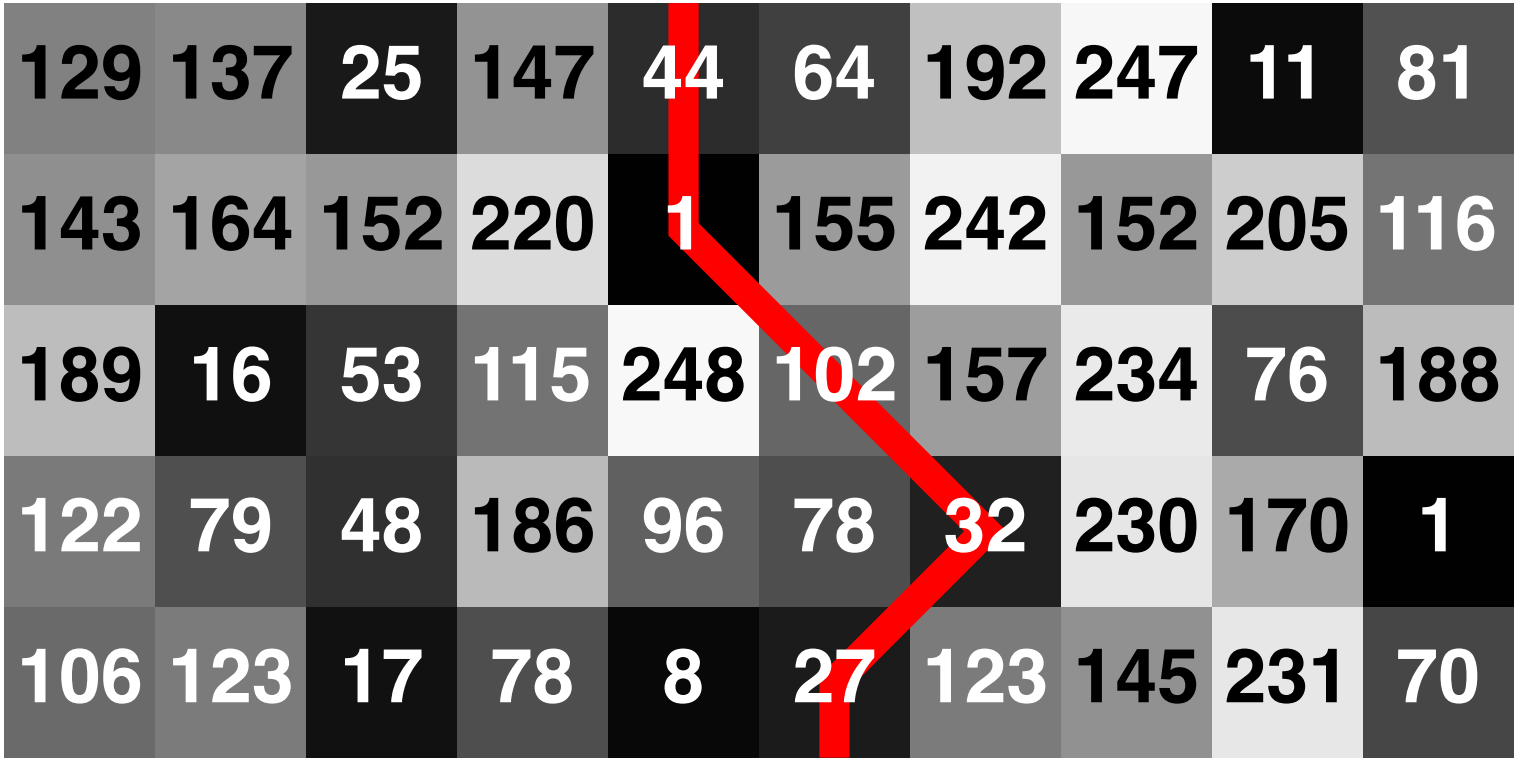

```python
def seam_carving(img_path: str, num_cols: int) -> None:
    ...

    for i in range(num_cols):
        seam = find_seam(img_grey)

        img_highlight = highlight_seam(img, seam)
        save_image(img_highlight, folder + f"highlight_{i}" + ext)
        img_grey_highlight = highlight_seam(img_grey, seam)
        save_image(img_grey_highlight, folder + f"highlight_{i}_grey" + ext)

        img = remove_seam(img, seam)
        save_image(img, folder + f"step_{i}" + ext)
        img_grey = remove_seam(img_grey, seam)
```

2. Recherche du seam de moindre énergie

129	137	25	147	44	64	192	247	11	81
143	164	152	220	1	155	242	152	205	116
189	16	53	115	248	102	157	234	76	188
122	79	48	186	96	78	32	230	170	1
106	123	17	78	8	27	123	145	231	70

Chaque cellule représente un pixel suite à l'application des différents filtres. Pour créer un seam : on part du haut et on descend soit tout droit, soit en allant d'un pixel sur la gauche, soit en allant d'un pixel sur la droite. **Le seam de moindre énergie est celui dont la somme des intensités est la plus faible.**

2. Recherche du seam de moindre énergie



Coût du seam de moindre énergie : 206.

- Comment le trouver ?

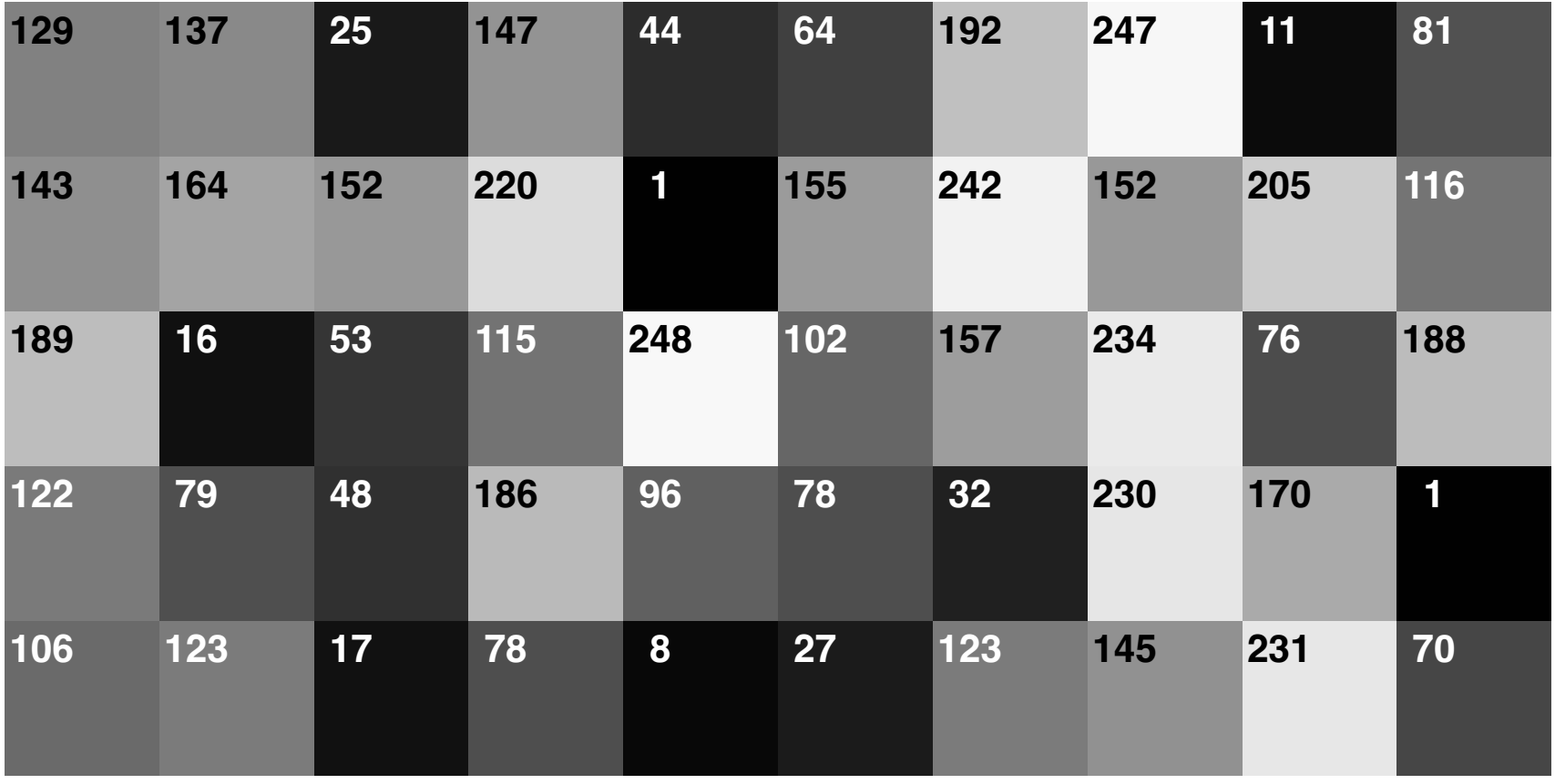
2. Recherche du seam de moindre énergie

Variante de l'algorithme du plus court chemin (Dijkstra)

- Idée de l'algorithme :
  - Le coût de chaque pixel de la première ligne est égal à son intensité.
  - Pour chaque ligne suivante :
    - Le coût d'un pixel est calculé en sommant son intensité avec le coût des trois pixels se situant sur la ligne précédente et en ne gardant que le coût minimum trouvé.
    - On garde en mémoire le pixel qui a conduit à ce coût minimal.
  - On cherche le pixel avec le coût le plus faible en dernière ligne, et on remonte «pixel après pixel» pour déterminer le seam de moindre énergie

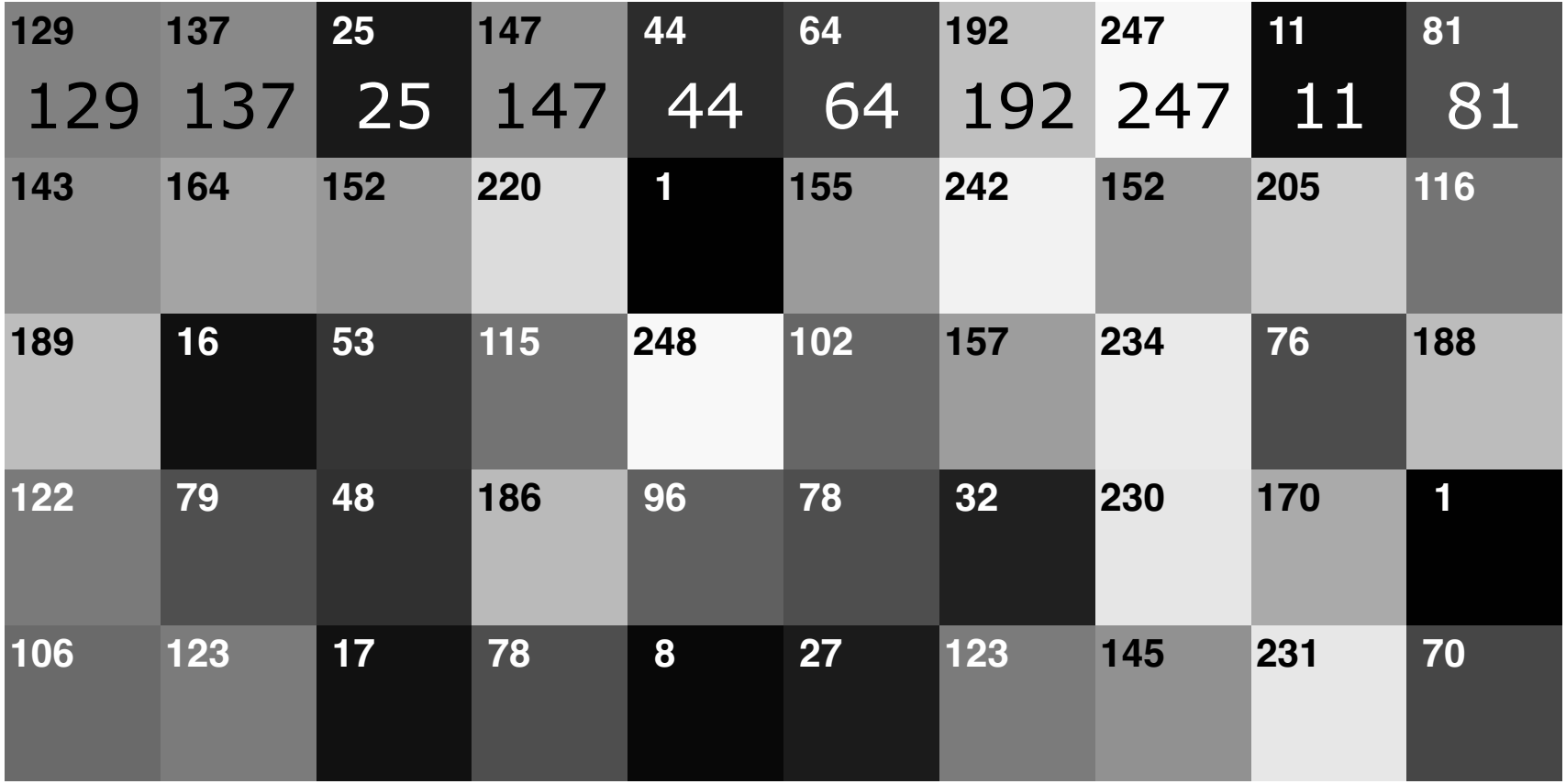
2. Recherche du seam de moindre énergie

Illustration



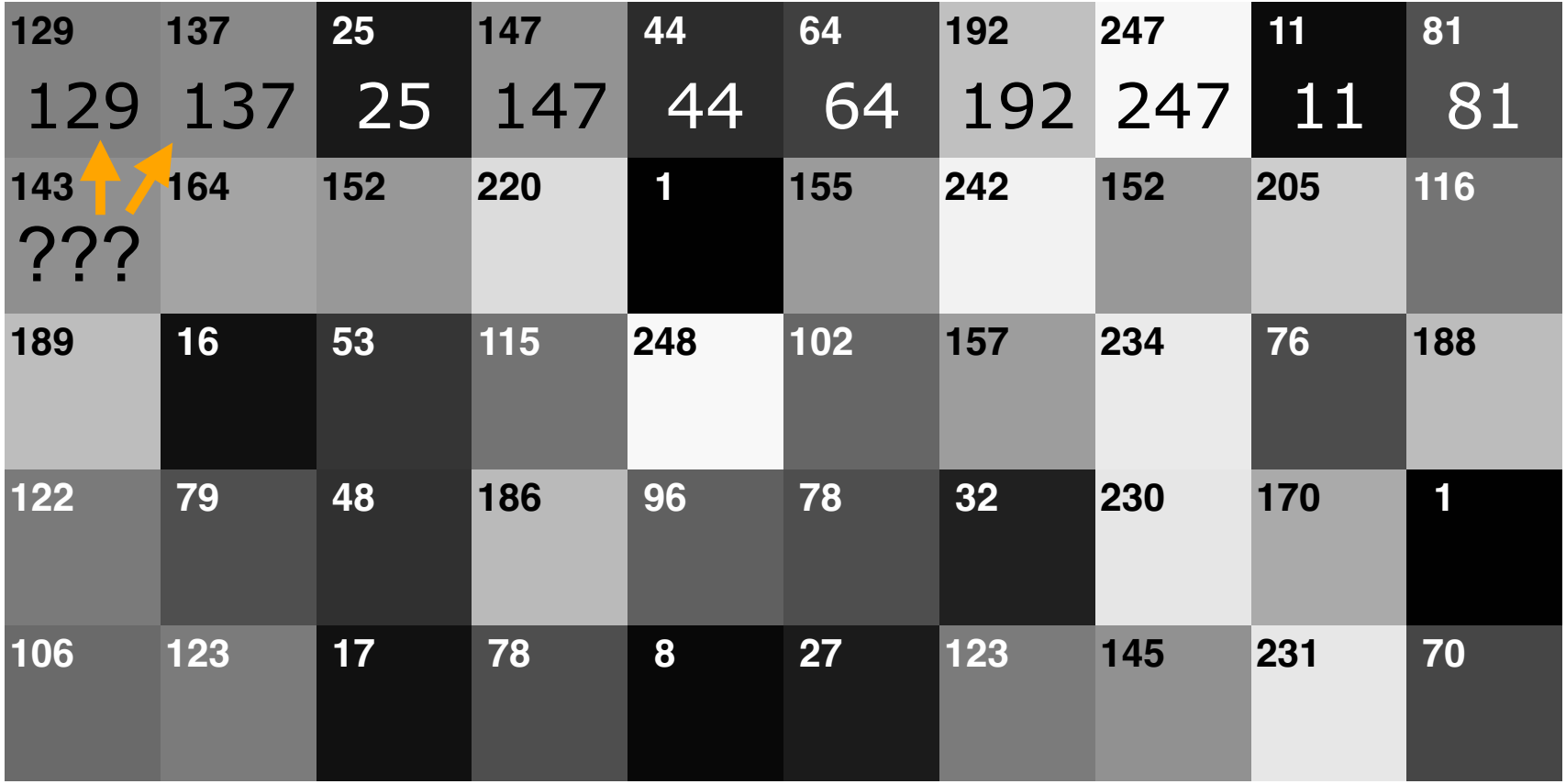
2. Recherche du seam de moindre énergie

Illustration



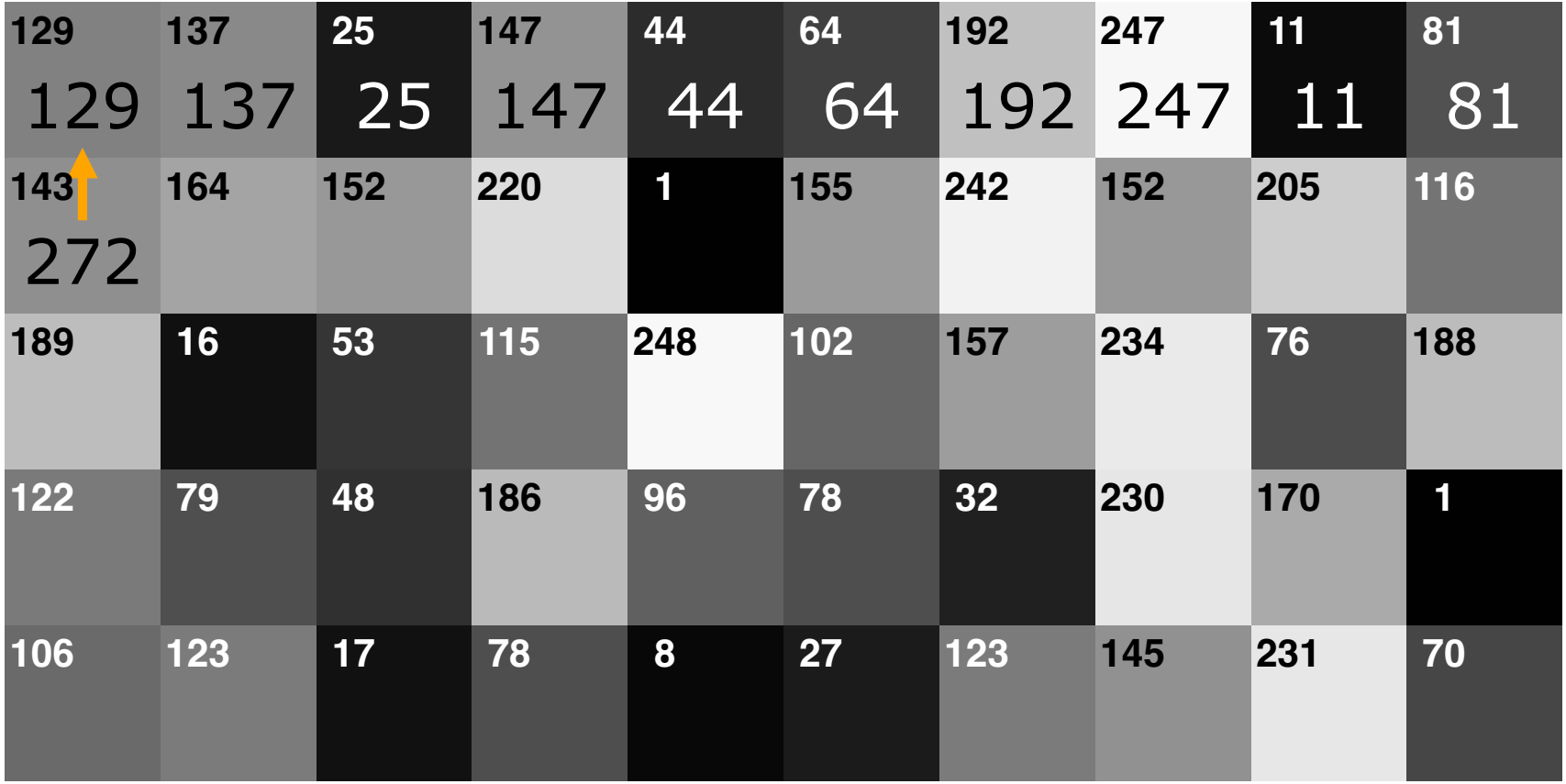
2. Recherche du seam de moindre énergie

Illustration



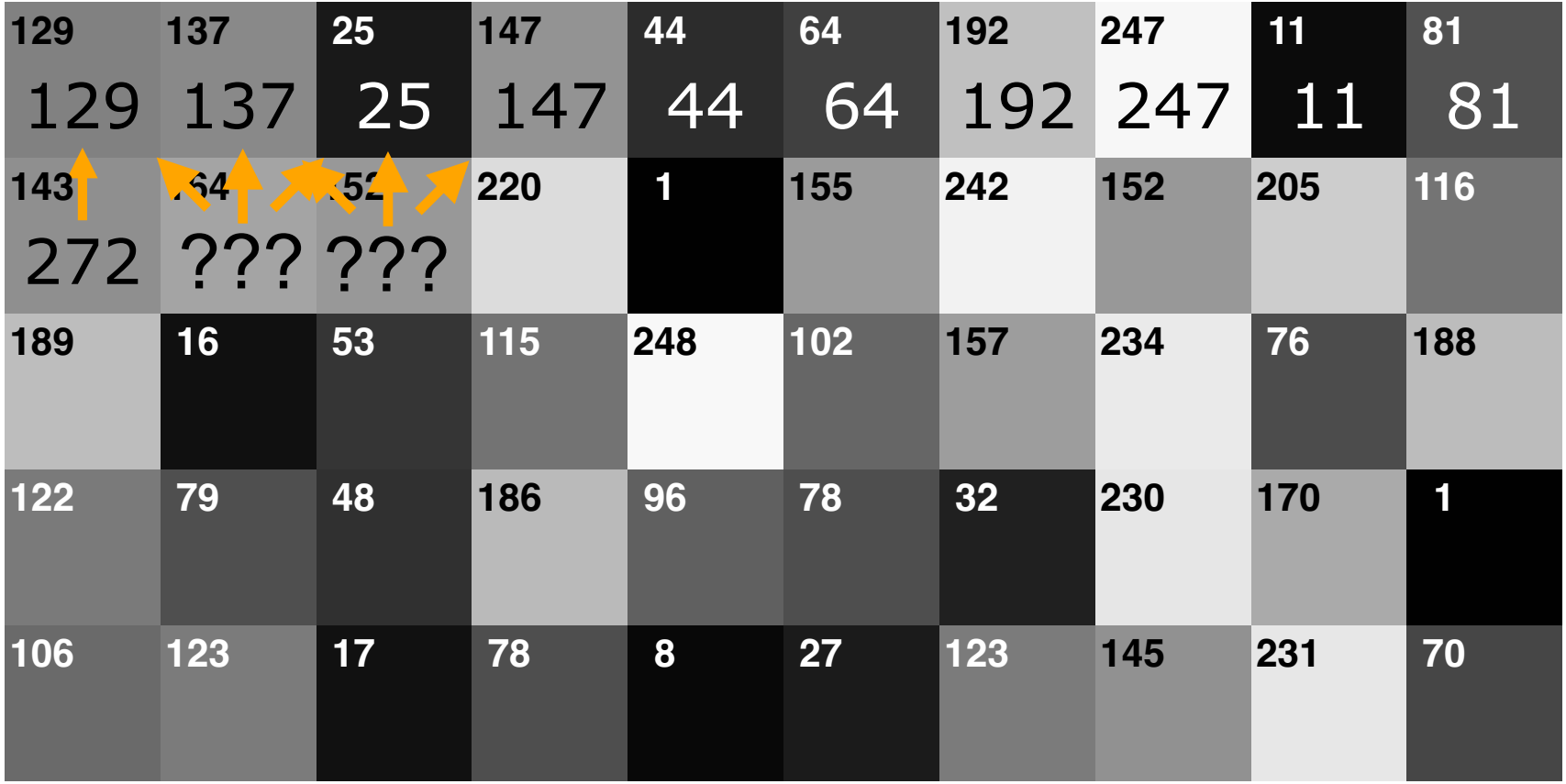
2. Recherche du seam de moindre énergie

Illustration



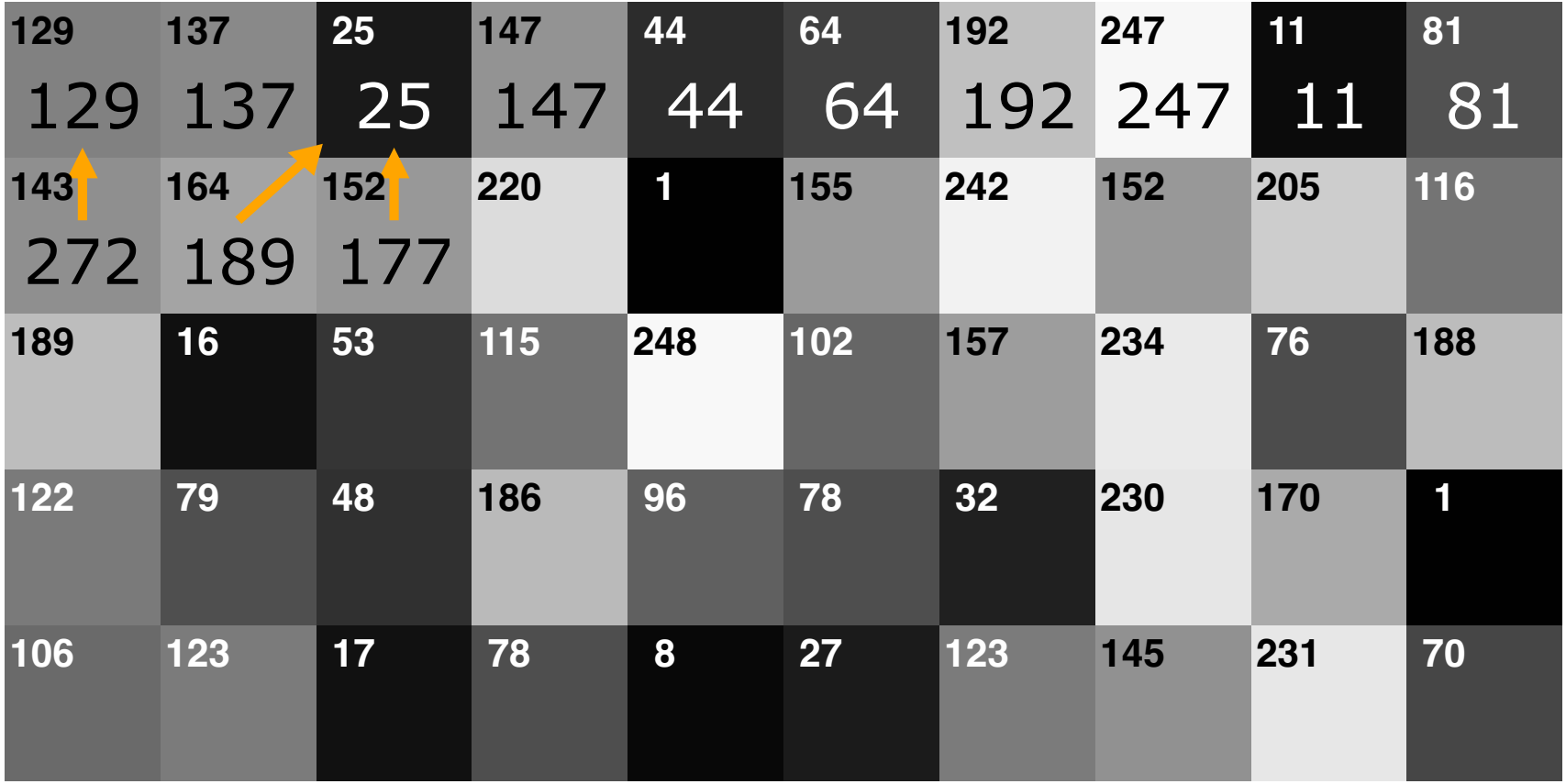
2. Recherche du seam de moindre énergie

Illustration



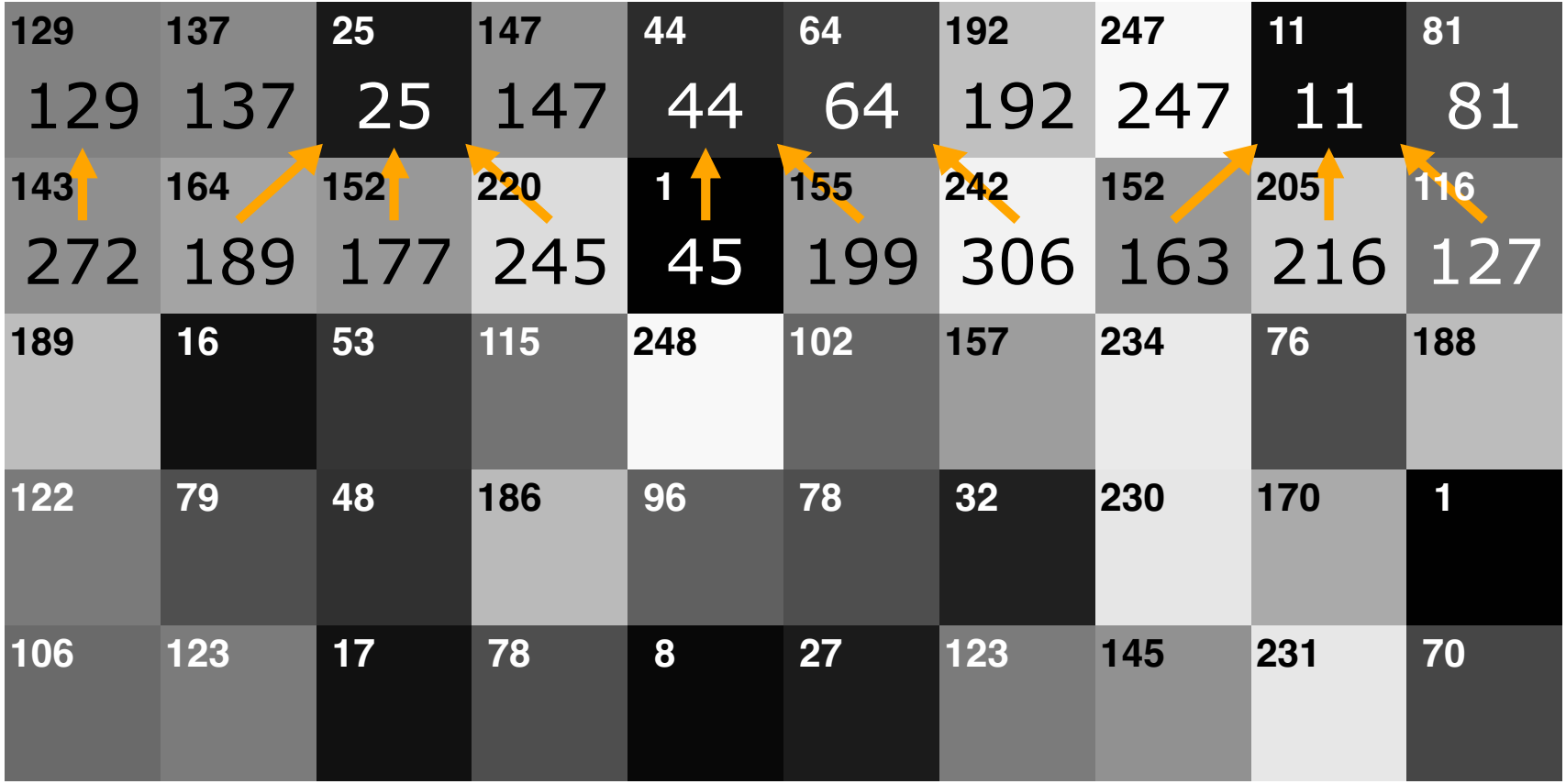
2. Recherche du seam de moindre énergie

Illustration



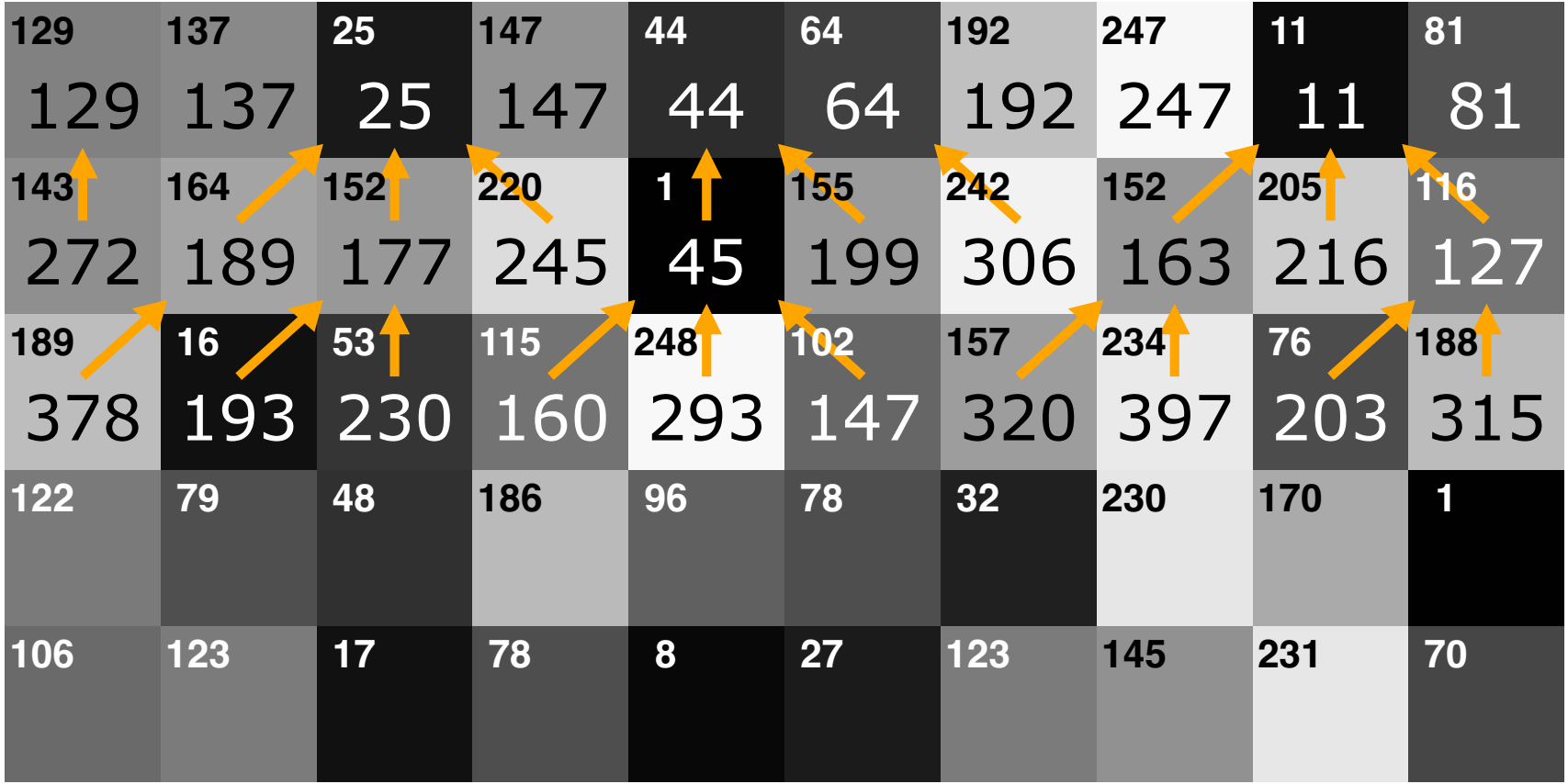
2. Recherche du seam de moindre énergie

Illustration



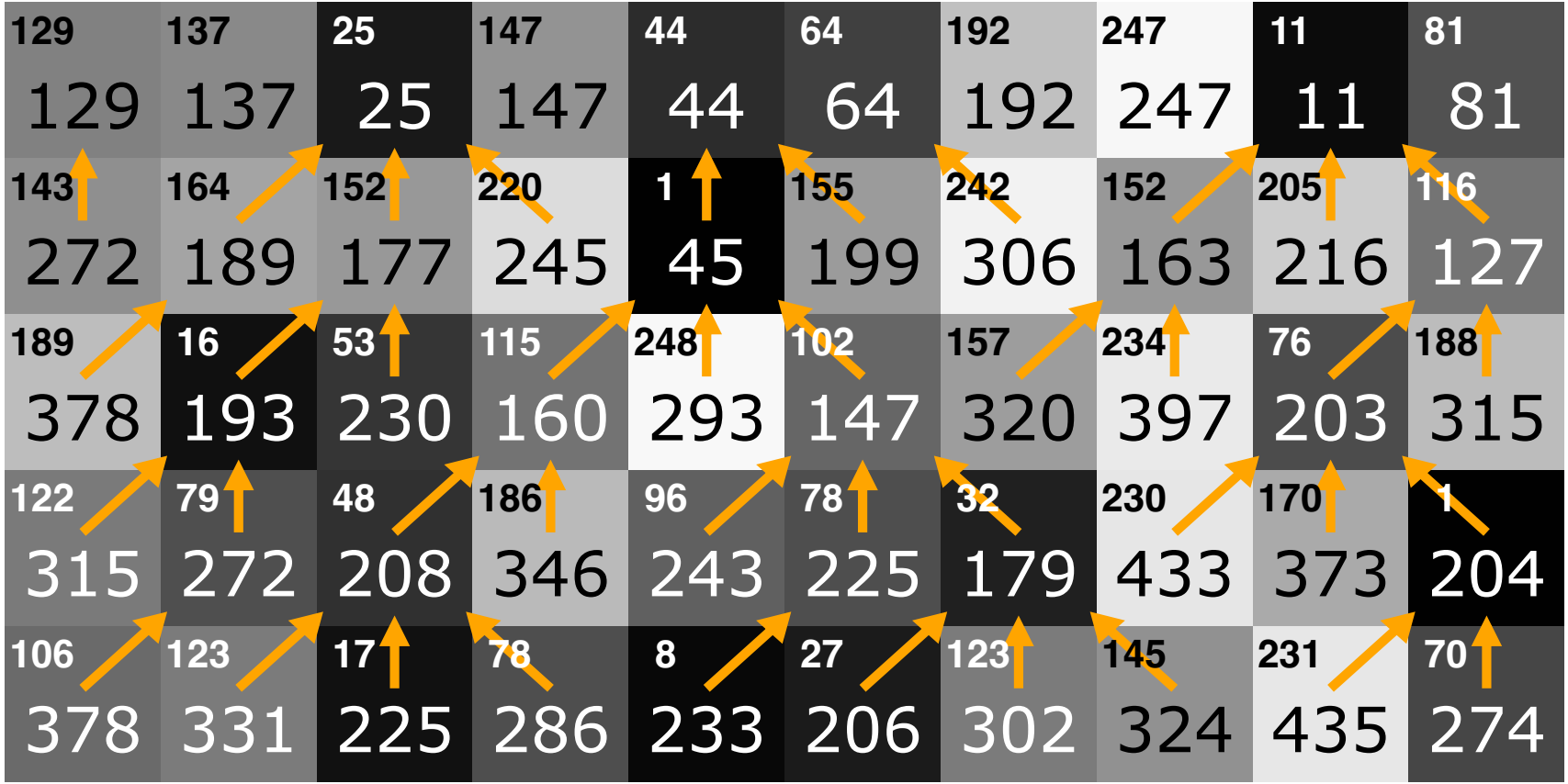
2. Recherche du seam de moindre énergie

Illustration



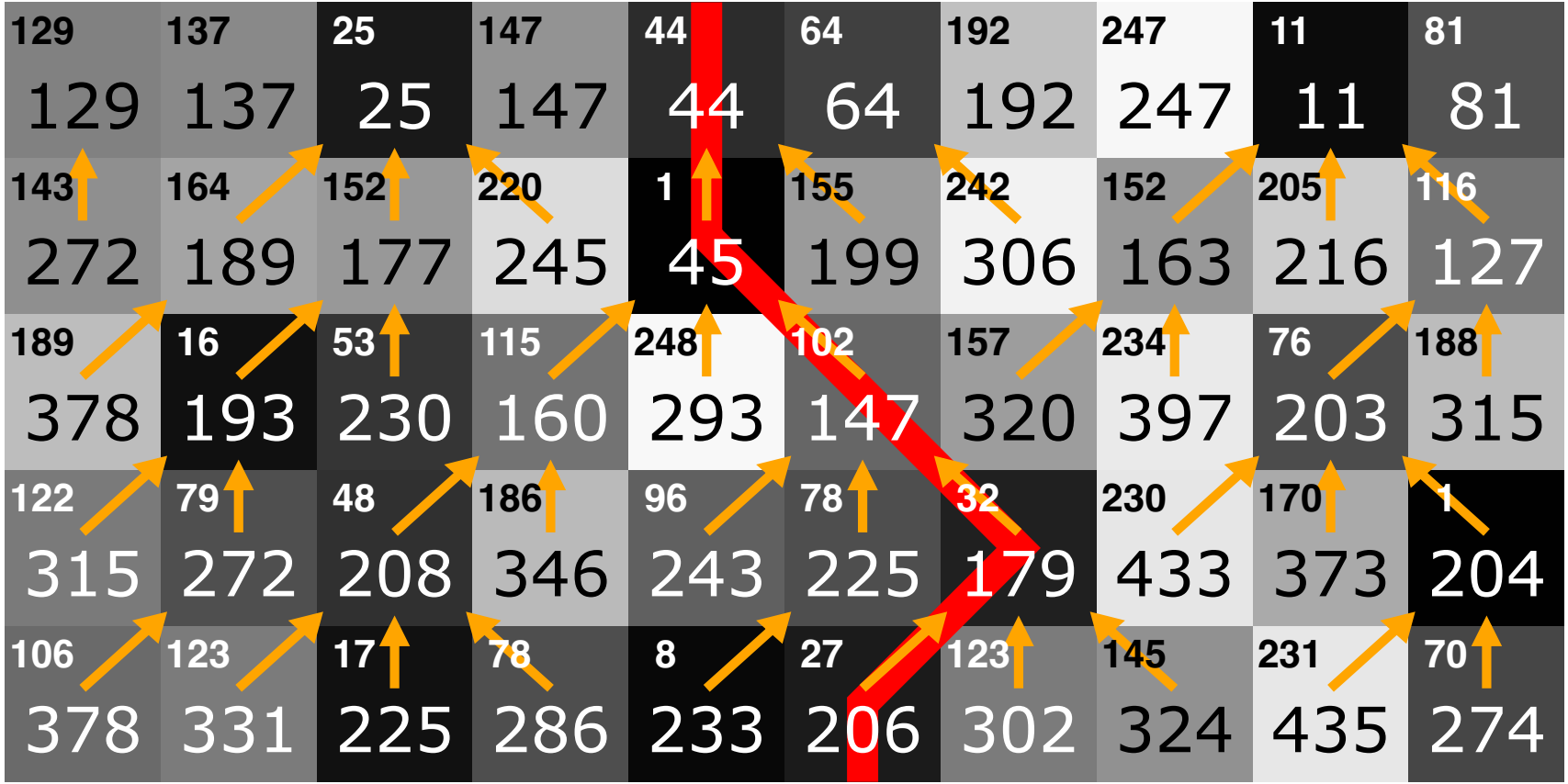
2. Recherche du seam de moindre énergie

Illustration



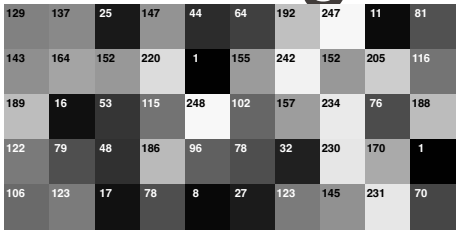
2. Recherche du seam de moindre énergie

Illustration



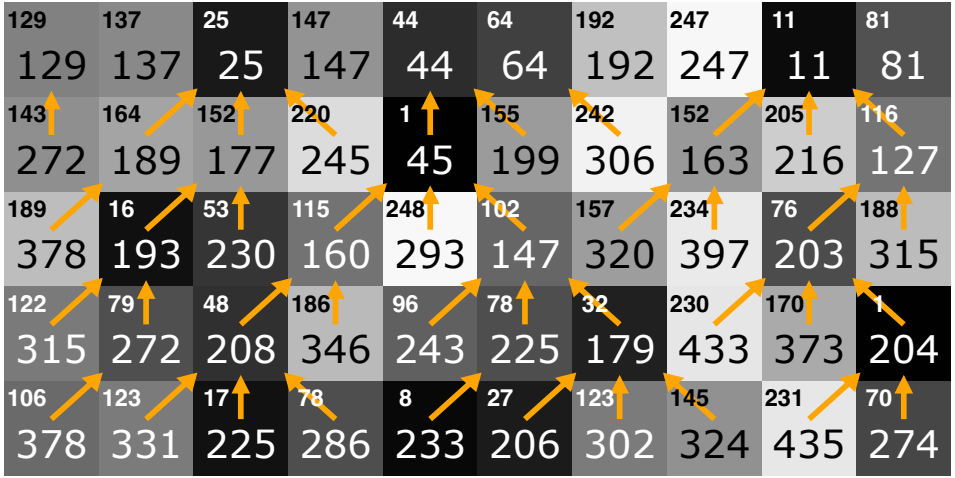
2. Recherche du seam de moindre énergie

Modélisation



- Vous avez une image avec des valeurs de gris (suite à la conversion en niveaux de gris et à l'application des filtres de lissage et de Sobel)
- Chaque pixel comporte deux données supplémentaires :
  - Le coût
  - Le parent

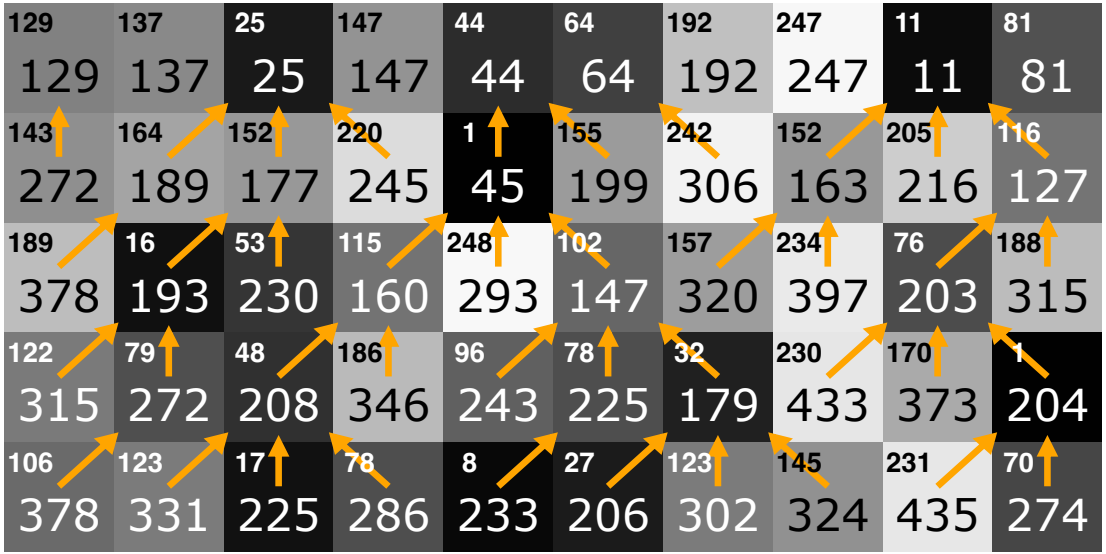
```
@dataclass
class PixelData:
    min_energy: int
    parent: Position
```



2. Recherche du seam de moindre énergie

Modélisation

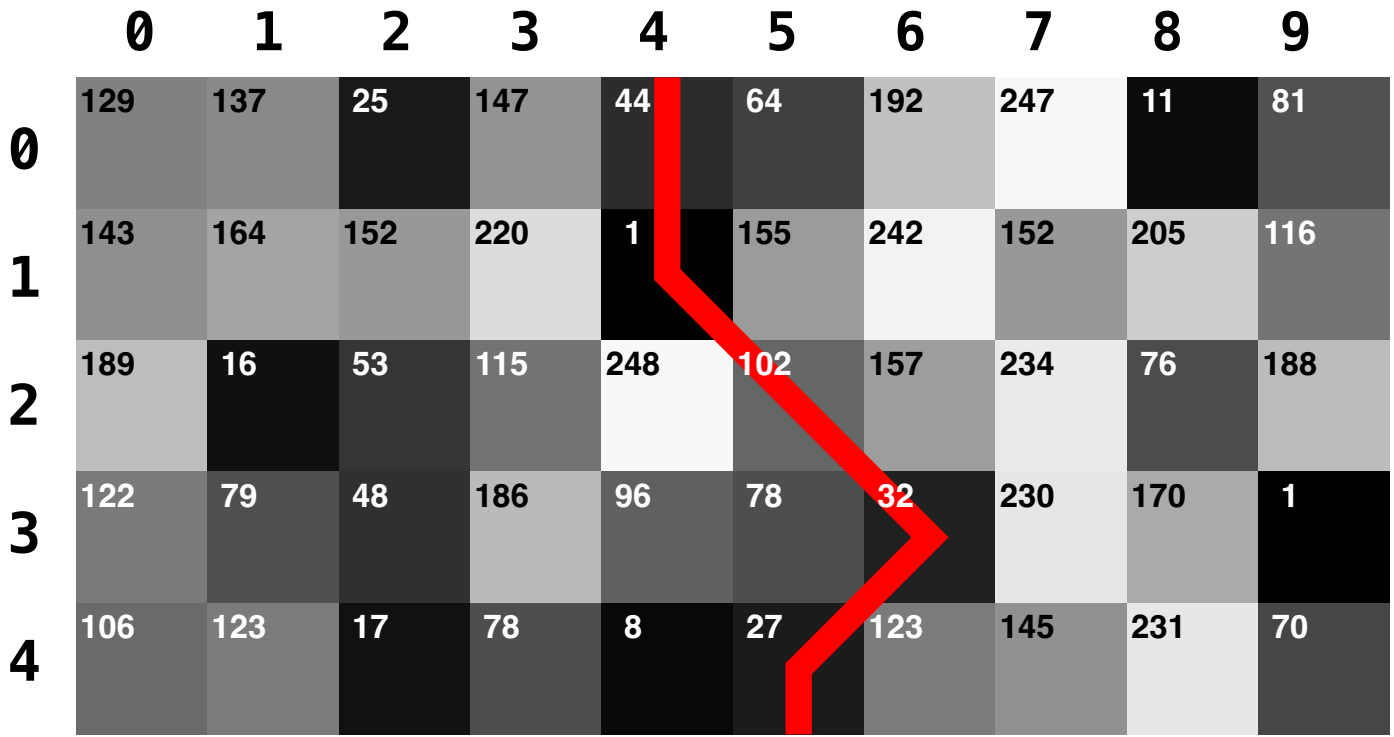
`List[List[PixelData]]`




```python
@dataclass
class PixelData:
    min_energy: int = Infinity
    parent: Position = (-1, -1)
```

2. Recherche du seam de moindre énergie

Modélisation



Seam trouvé :
`[4, 4, 5, 6, 5]`

Petite démo

2. Recherche du seam de moindre énergie

Nouvelles fonctions dans miniprojectutils.py

- `def save_image_greyscale_details(img: Image, filename: str, seam: Seam):`
  - Sauvegarde une image en niveaux de gris avec les valeurs écrites dessus (comme dans le cours)
  - Peut aussi afficher le seam s'il est passé en paramètre
- `def new_image_grey_with_data(data: list[list[int]]) -> Image:`
  - Permet de créer une nouvelle image en niveaux de gris, où data représente une matrice d'entiers compris entre 0 et 255
- `def new_random_grey_image(height: int, width: int) -> Image:`
  - Permet de créer aléatoirement une image en niveaux de gris
  - Utile pour tester votre fonction de recherche de seam

Bilan

- Deuxième et dernière séance d'exercices consacrées au mini-projet
- Instructions pour le rendu qui arriveront en début de semaine prochaine
- Rendu fixé au dimanche 10 décembre 2023, 23h59

- Points de vigilance :
  - Prenez garde à ne pas modifier l'image d'origine ! Dans le doute, téléchargez encore une fois les images depuis Moodle.
  - Lorsque vous créerez une `List[List[PixelData]]`, assurez-vous d'instancier autant de fois que nécessaire `PixelData()` et à ne pas en dupliquer qu'un seul. Tous vont alors pointer vers le même objet.