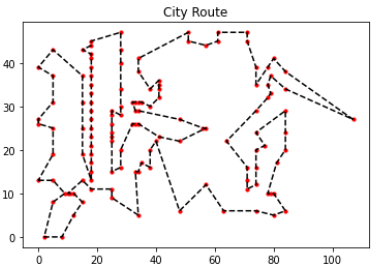
인공지능 Summative Assignment 4

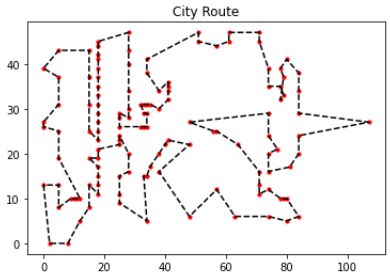
20193494 문서형

Simulated Annealing 이 확률을 이용하는 무작위성이 있어서 각 Cooling Ratio 에 대해서 결과를 3 번 도출하고 평균 값을 이용하겠습니다.

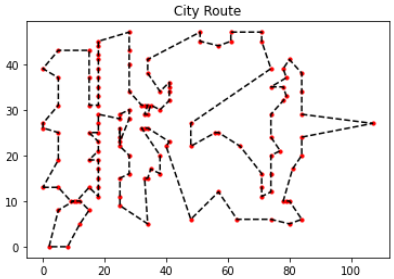
1. 기본으로 설정되어 있는 T = 0.5 * T




```
Execution Time: 9.808867692947388
Path: [0, 5, 13, 11, 4, 12, 17, 24, 16, 15, 14, 18, 25, 44, 52, 73, 63, 67, 74, 76, 77, 80, 88, 92, 97, 111, 122, 129, 120, 1
17, 113, 123, 124, 125, 126, 107, 112, 106, 105, 104, 99, 100, 101, 98, 108, 114, 118, 115, 119, 127, 130, 128, 121, 116, 10
9, 110, 102, 103, 96, 95, 94, 89, 90, 72, 71, 79, 85, 84, 83, 82, 78, 75, 70, 66, 62, 65, 69, 87, 91, 93, 86, 81, 68, 64, 61,
54, 53, 45, 46, 47, 48, 49, 50, 51, 55, 56, 57, 58, 59, 60, 43, 42, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 2
8, 27, 26, 19, 20, 21, 22, 10, 3, 9, 8, 2, 1, 7, 6, 0]
Cost: 630.95322144724
```




```
Execution Time: 9.83948826789856
Path: [0, 4, 12, 17, 24, 18, 25, 26, 27, 28, 19, 29, 30, 46, 47, 48, 54, 53, 45, 44, 52, 73, 63, 67, 74, 77, 80, 81, 86, 76,
88, 92, 97, 111, 122, 129, 120, 117, 113, 104, 99, 100, 101, 98, 93, 91, 87, 108, 107, 112, 106, 105, 123, 124, 125, 130, 12
6, 127, 128, 121, 116, 119, 114, 118, 115, 109, 110, 102, 103, 96, 95, 94, 89, 90, 72, 71, 79, 85, 84, 83, 82, 78, 75, 70, 6
6, 62, 65, 69, 68, 64, 61, 49, 50, 51, 55, 56, 57, 58, 59, 60, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 20, 21, 2
2, 23, 10, 3, 9, 8, 2, 1, 7, 6, 16, 15, 14, 13, 11, 5, 0]
Cost: 617.8481236827581
```

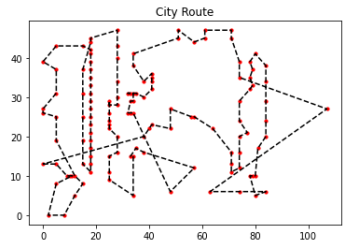



```
Execution Time: 9.95389986038208
Path: [0, 5, 13, 14, 11, 4, 12, 17, 24, 16, 15, 18, 25, 26, 27, 28, 29, 19, 30, 31, 20, 32, 33, 34, 50, 51, 56, 55, 47, 48, 4
9, 46, 54, 53, 45, 44, 52, 73, 63, 67, 74, 76, 77, 61, 64, 68, 81, 80, 88, 92, 97, 111, 122, 129, 120, 117, 113, 123, 124, 12
5, 130, 126, 127, 128, 121, 116, 119, 109, 115, 118, 114, 108, 107, 112, 106, 105, 104, 99, 100, 101, 98, 93, 91, 86, 87, 11
0, 102, 103, 96, 95, 94, 89, 90, 72, 71, 79, 85, 84, 83, 82, 78, 75, 69, 65, 70, 66, 62, 57, 58, 59, 60, 43, 42, 41, 40, 39,
38, 37, 36, 35, 21, 22, 23, 10, 3, 9, 8, 2, 1, 7, 6, 0]
Cost: 621.2598682459146
```

Execution Time : 9.86 Cost: 623.34

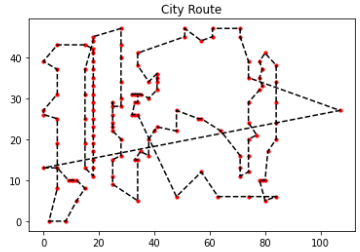
2. Initial Temperature 변화

Initial_T = 1000



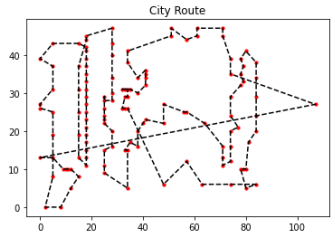

```
Execution Time: 9.583700180053711
Path: [0, 5, 13, 14, 15, 11, 4, 12, 17, 24, 16, 6, 7, 1, 2, 8, 9, 3, 10, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
29, 28, 27, 26, 25, 18, 19, 20, 21, 22, 42, 43, 60, 59, 58, 57, 56, 55, 50, 51, 49, 48, 47, 46, 54, 53, 45, 44, 52, 73, 67, 6
3, 74, 76, 92, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79, 71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 10
9, 130, 97, 111, 129, 122, 113, 117, 120, 123, 124, 125, 126, 127, 128, 121, 116, 119, 115, 118, 114, 108, 107, 112, 106, 10
5, 104, 99, 100, 101, 98, 93, 91, 87, 86, 81, 80, 77, 0]
Cost: 679.9285783186399
```

Initial_T = 10000




```
Execution Time: 10.207612037658691
Path: [0, 5, 13, 14, 15, 16, 24, 17, 12, 4, 11, 6, 7, 1, 2, 8, 9, 3, 10, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
29, 28, 27, 26, 25, 18, 19, 20, 21, 22, 42, 43, 60, 59, 58, 57, 56, 55, 50, 51, 49, 48, 47, 46, 54, 53, 45, 44, 52, 73, 67, 6
3, 74, 76, 77, 80, 81, 86, 87, 91, 93, 98, 101, 100, 99, 104, 105, 106, 112, 107, 108, 114, 118, 115, 119, 116, 121, 128, 12
7, 126, 125, 124, 123, 120, 117, 113, 122, 129, 111, 97, 92, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79,
71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 109, 130, 0]
Cost: 687.7892183230354
```

Initial_T = 100000

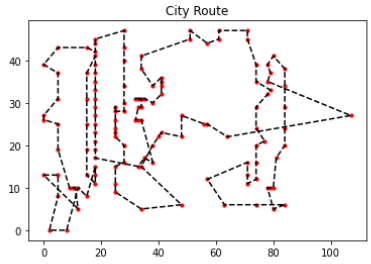



```
Execution Time: 10.419121265411377
Path: [0, 5, 13, 14, 15, 16, 24, 17, 12, 4, 11, 6, 7, 1, 2, 8, 9, 3, 10, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
29, 28, 27, 26, 25, 18, 19, 20, 21, 22, 42, 43, 60, 59, 58, 57, 56, 55, 50, 51, 49, 48, 47, 46, 54, 53, 45, 44, 52, 73, 67, 6
3, 74, 76, 77, 80, 81, 86, 87, 91, 93, 98, 101, 100, 99, 104, 105, 106, 112, 107, 108, 114, 118, 115, 119, 116, 121, 128, 12
7, 126, 125, 124, 123, 120, 117, 113, 122, 129, 111, 97, 92, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79,
71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 109, 130, 0]
Cost: 687.7892183230354
```

확실히 초기 T 를 높일수록 Execution Time 이 늘어나는 것을 확인할 수 있었습니다. Cost 는 기존 100 일 때에 비해서 오히려 더 안 좋아진 모습을 볼 수 있습니다. 높은 Temperature 로 인해 많은 Bad Move 를 Accept 해서 max optimize value 와는 멀어졌기 때문에 더 안 좋은 결과를 보인 것으로 보입니다.

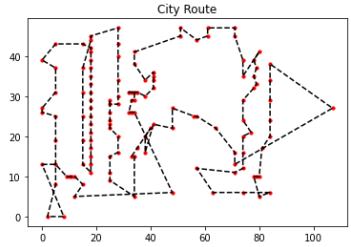
Cooling Ratio 변화

Cooling Ratio = 0.1



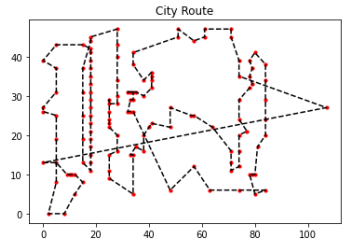

```
Execution Time: 9.09625506401062
Path: [0, 5, 11, 4, 12, 16, 24, 27, 26, 25, 18, 19, 20, 21, 22, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 63, 74, 7
6, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79, 71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 109, 118, 114, 108,
107, 112, 106, 105, 104, 99, 100, 101, 92, 97, 111, 129, 122, 113, 117, 120, 123, 124, 125, 126, 127, 128, 121, 116, 119, 11
5, 130, 98, 93, 91, 87, 86, 81, 80, 77, 67, 88, 73, 52, 44, 45, 53, 54, 46, 47, 48, 49, 51, 50, 55, 56, 57, 58, 59, 60, 43, 4
2, 41, 23, 10, 3, 9, 8, 2, 1, 7, 6, 13, 14, 15, 17, 0]
Cost: 662.034142827834
```

Cooling Ratio = 0.3




```
Execution Time: 9.351080417633057
Path: [0, 5, 13, 14, 15, 16, 24, 17, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79, 71, 72, 89, 90, 94, 95,
96, 103, 102, 110, 109, 121, 116, 119, 115, 118, 114, 108, 107, 112, 106, 105, 104, 99, 92, 97, 111, 129, 122, 113, 117, 120,
123, 124, 125, 126, 127, 128, 130, 100, 101, 98, 93, 91, 87, 86, 81, 76, 77, 80, 74, 63, 67, 73, 52, 44, 45, 53, 54, 46, 47,
48, 49, 51, 50, 55, 56, 57, 58, 59, 60, 43, 42, 22, 21, 20, 19, 18, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3
9, 40, 41, 23, 10, 3, 9, 8, 2, 1, 7, 6, 11, 4, 12, 0]
Cost: 669.7326224302361
```

Cooling Ratio = 0.7

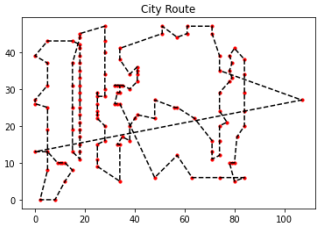



```
Execution Time: 10.83526086807251
Path: [0, 5, 13, 14, 15, 16, 24, 17, 12, 4, 11, 6, 7, 1, 2, 8, 9, 3, 10, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
29, 28, 27, 26, 25, 18, 19, 20, 21, 22, 42, 43, 60, 59, 58, 57, 56, 55, 50, 51, 49, 48, 47, 46, 54, 53, 45, 44, 52, 73, 67, 6
3, 74, 76, 77, 80, 81, 86, 87, 91, 93, 98, 101, 100, 99, 104, 105, 106, 112, 107, 108, 114, 118, 115, 119, 116, 121, 128, 12
7, 126, 125, 124, 123, 120, 117, 113, 122, 129, 111, 97, 92, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79,
71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 109, 130, 0]
Cost: 687.7892183230354
```

Cooling Ratio = 0.9

확실히 Cooling Ratio 가 높을수록, Temperature 이 완만하게 감소할수록 Execution Time 이 높은 것을 확인할 수 있습니다. Temperature 이 높으면 Bad Move 를 Accept 할 확률이 높아지기 때문에 반복횟수가 높아 Execution time 이 늘어나는 것으로 해석됩니다.

Cost 는 0.1, 0.3, 0.5, 0.7, 0.9 의 Cooling Ratio 의 결과를 비교해봤을 때 0.5 에서 최솟값을 가지고 이외의 경우에는 Cooling Ratio 가 높았습니다. 이는 0.5 가 다른 Cooling Ratio 에 비해 적절한 양의 Bad Move 를




```
Execution Time: 14.681533336639404
Path: [0, 5, 13, 14, 15, 16, 24, 17, 12, 4, 11, 6, 7, 1, 2, 8, 9, 3, 10, 23, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30,
29, 28, 27, 26, 25, 18, 19, 20, 21, 22, 42, 43, 60, 59, 58, 57, 56, 55, 50, 51, 49, 48, 47, 46, 54, 53, 45, 44, 52, 73, 67, 6
3, 74, 76, 77, 80, 81, 86, 87, 91, 93, 98, 101, 100, 99, 104, 105, 106, 112, 107, 108, 114, 118, 115, 119, 116, 121, 128, 12
7, 126, 125, 124, 123, 120, 117, 113, 122, 129, 111, 97, 92, 88, 68, 64, 61, 65, 69, 70, 66, 62, 75, 78, 82, 83, 84, 85, 79,
71, 72, 89, 90, 94, 95, 96, 103, 102, 110, 109, 130, 0]
Cost: 687.7892183230354
```

accept 해 최적값에서 멈출 수 있었던 것으로 보입니다.

3. 개선안 제시

Formative Assignment 에서 제출한 Cooling Ratio 아이디어입니다.

t 가 작을 때는 여러가지 탐색을 해보다가 일정 시간 이후 T 를 줄여 탐색을 할 수 있으면 좋겠다는 생각을 했습니다.

T 가 너무 급격하게 줄어들면 최적 값을 가지는 영역에 들어가기 전에 local optima 에 다다를 것 같다는 생각을 했고 값이 커질수록 값이 급격히 감소하는 log(-t) 함수의 특성에 착안해 만 들어 보았습니다. 상수 a, b 를 조절해 T 의 감소 속도를 조절할 수 있습니다.

기존 0.5 를 곱하는 함수를 그래프로 그렸을 때 빨간색 선입니다.

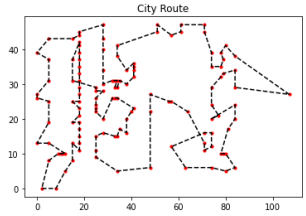
개선안의 log(-0.2t + 4)를 곱하는 그래프를 그렸을 때 초록색입니다. 제가 기대한 효과는 다음과 같습니다. :

t 가 낮을 때는 기존보다 T 가 크고 이후로 갈수록 빠르게 떨어지기 때문에 높은 Temperature 로 더 확실하게 Bad Move 를 Accept 합니다. t 가 일정 이상 높아지면 기존보다 Temperature 가 빠르게 감소해서 기존 보다 빠르게 Optimization 이 진행됩니다.

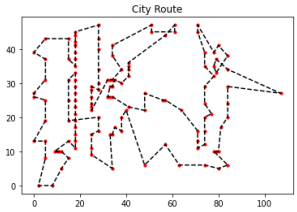
4. 개선안 구현 및 실행

```
def Cooling_Ratio(T, t):
    return T * (math.log10(-0.2*t + 4))
```

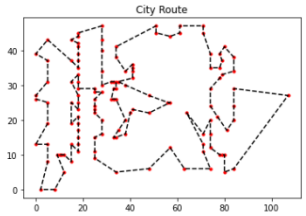
```
temperature = TEMPERATURE
t = 0
while temperature > TEMP_LIMIT:
    curr_path = best_path.copy()
    # Select two indices of flip points
    sel_idx = np.sort(np.random.choice(np.arange(1, cnt_cities + 1), 2))

    # Path Flip and update cost array
    curr_path[sel_idx[0]:sel_idx[1]] = np.flip(curr_path[sel_idx[0]: sel_idx[1]])
    cost_arr = path_cost(path_map, curr_path)
    curr_cost = cost_arr.sum()

    if curr_cost <= best_cost:
        best_path, best_cost = curr_path, curr_cost
    else:
        prob = 1 / np.exp((curr_cost - best_cost) / float(temperature))
        if prob > np.random.rand(1):
            best_path, best_cost = curr_path, curr_cost
    temperature = Cooling_Ratio(temperature, t)
    t += 1
return best_path, best_cost
```




```
Execution Time: 9.777893304824829
Path: [0, 5, 16, 15, 14, 13, 11, 4, 12, 17, 24, 18, 25, 26, 27, 28, 29, 19, 30, 31, 20, 32, 33, 34, 35, 36, 37, 38, 39, 40, 4
1, 22, 21, 51, 50, 55, 49, 48, 47, 46, 54, 61, 64, 68, 81, 80, 77, 76, 74, 67, 63, 53, 45, 44, 52, 73, 88, 86, 87, 91, 93, 9
8, 100, 99, 104, 105, 101, 92, 97, 111, 122, 129, 120, 117, 113, 123, 124, 125, 106, 112, 107, 108, 114, 118, 127, 126, 130,
128, 121, 116, 119, 115, 109, 110, 102, 103, 96, 95, 94, 90, 89, 72, 71, 79, 85, 84, 83, 82, 78, 75, 69, 65, 70, 66, 62, 56,
57, 58, 59, 60, 43, 42, 23, 10, 3, 9, 8, 2, 1, 7, 6, 0]
Cost: 609.8019367880004
```




```
Execution Time: 9.707057476043701
Path: [0, 6, 7, 1, 2, 8, 9, 3, 10, 23, 22, 37, 38, 39, 40, 41, 42, 43, 60, 59, 58, 57, 56, 55, 51, 50, 49, 48, 47, 46, 62, 6
6, 70, 79, 71, 72, 90, 89, 95, 96, 94, 85, 84, 83, 82, 78, 75, 69, 65, 61, 64, 68, 81, 86, 87, 91, 93, 98, 101, 100, 99, 104,
105, 106, 112, 107, 108, 114, 109, 110, 102, 103, 116, 121, 128, 118, 115, 119, 127, 130, 126, 125, 124, 123, 120, 117, 113,
129, 122, 111, 97, 92, 88, 80, 77, 76, 74, 67, 63, 73, 52, 44, 45, 53, 54, 19, 20, 21, 36, 35, 34, 33, 32, 31, 30, 29, 28, 2
7, 26, 25, 18, 13, 14, 15, 16, 24, 17, 12, 4, 11, 5, 0]
Cost: 631.1745286093847
```




```
Execution Time: 9.77042841911316
Path: [0, 6, 7, 1, 2, 8, 9, 3, 10, 22, 37, 38, 39, 40, 41, 23, 42, 43, 60, 59, 58, 57, 56, 62, 66, 70, 69, 65, 61, 64, 68, 7
7, 74, 63, 67, 76, 80, 81, 86, 93, 91, 87, 78, 75, 82, 83, 84, 85, 79, 71, 72, 90, 89, 94, 95, 96, 103, 102, 110, 109, 115, 1
19, 116, 121, 128, 127, 118, 114, 108, 107, 112, 123, 124, 125, 126, 130, 129, 122, 120, 117, 113, 104, 105, 106, 101, 98, 10
0, 99, 111, 97, 92, 88, 73, 52, 44, 45, 53, 54, 46, 47, 48, 49, 55, 50, 51, 34, 35, 36, 21, 33, 32, 31, 20, 19, 30, 29, 28, 2
7, 26, 25, 18, 24, 16, 15, 14, 13, 17, 12, 4, 11, 5, 0]
Cost: 611.6145708200297
```

Execution Time: 9.74 Cost: 617.52

Execution Time 과 Cost 모두 기존에 비해서 나아진 모습을 볼 수 있었습니다.

제가 설계할 때 Execution Time 은 확실히 줄어들 것이라고 예상했습니다. t 가 4 이상일 때 기존 보다 낮은 T 를 가지므로 빠르게 멈출 것으로 생각했는데 맞아 떨어졌지만 생각보다 아주 큰 차이는 아니었습니다. T 를 바꿔도 결국은 accept 할지는 확률이기 때문에 그런 것일까 추측해봅니다.

Cost 의 경우 확실히 예상하지 못했습니다. Bad Move 를 많이 혹은 적게 Accept 한다고 무조건 좋아지는 것이 아니라 적절히 Accept 해야 좋은 Cost 를 얻을 수 있는데 제가 고안한 Cooling Ratio 가 적절한지 예상하기 힘들었던 것 같습니다. 결과를 보니 기존보다 조금이지만 좋게 나와서 만족합니다.