동국대학교

Lab 2 알고리즘 레포트

|  |  |
| --- | --- |
| 이름 | 한윤수 |
| 학과 | 컴퓨터AI학부 |
| 학번 | 2021113405 |
| 제출일자 | 2025.09.18 |

문제 1. 임의의 자연수 N을 입력 받아서 N개의 랜덤한 데이터를 생성해 반복문과 조건문 만을 이용하여 삽입정렬의 각 단계별 결과를 보여주는 코드를 작성하고 반드시 코드에 대한 자세한 설명을 추가하시오.

입력 예제

생성할 랜덤 변수 개수를 입력하세요: 10

출력 예제

생성된 list : [67, 14, 27, 8, 56, 18, 3, 96, 15, 96]

삽입 정렬 시작:

단계별 선택된 key : 14

단계 1: [14, 67, 27, 8, 56, 18, 3, 96, 15, 96]

단계별 선택된 key : 27

단계 2: [14, 27, 67, 8, 56, 18, 3, 96, 15, 96]

단계별 선택된 key : 8

단계 3: [8, 14, 27, 67, 56, 18, 3, 96, 15, 96]

단계별 선택된 key : 56

단계 4: [8, 14, 27, 56, 67, 18, 3, 96, 15, 96]

단계별 선택된 key : 18

단계 5: [8, 14, 18, 27, 56, 67, 3, 96, 15, 96]

단계별 선택된 key : 3

단계 6: [3, 8, 14, 18, 27, 56, 67, 96, 15, 96]

단계별 선택된 key : 96

단계 7: [3, 8, 14, 18, 27, 56, 67, 96, 15, 96]

단계별 선택된 key : 15

단계 8: [3, 8, 14, 15, 18, 27, 56, 67, 96, 96]

단계별 선택된 key : 96

단계 9: [3, 8, 14, 15, 18, 27, 56, 67, 96, 96]

최종 정렬 결과: [3, 8, 14, 15, 18, 27, 56, 67, 96, 96]

소스코드 및 설명

|  |
| --- |
| import random  def main():      test = False      if test:          array\_size = 10          array = [67, 14, 27, 8, 56, 18, 3, 96, 15, 96]      else:          array\_size = int(input("생성할 랜덤 변수 개수를 입력하세요: "))          array = random.sample(range(1, 101), array\_size)      ans = []      ans.append("생성된 list : [" + ", ".join(map(str, array)) + "]")      ans.append("삽입 정렬 시작:")      for i in range(array\_size - 1):  # size - 1번 동안 삽입 정렬 실행          curr\_idx = i + 1  # 현재 key의 위치          key = array[curr\_idx]  # 삽입할 값 저장          j = i  # key 앞쪽 인덱스부터 검사 시작          ans.append(f"단계별 선택된 key : {key}")          # key보다 큰 값들을 한 칸씩 뒤로 이동          while j >= 0 and array[j] > key:              array[j + 1] = array[j]              j -= 1          # j가 멈춘 자리 바로 뒤에 key 삽입          array[j + 1] = key          ans.append(f"단계 {i + 1}:[" + ", ".join(map(str, array)) + "]")      ans.append("최종 정렬 결과: [" + ", ".join(map(str, array)) + "]")      final\_ans = "\n".join(ans)      print(final\_ans)  if \_\_name\_\_ == "\_\_main\_\_":      main() |

설명 - 삽입 정렬 과정:

1. 리스트의 두 번째 원소부터 시작하여 현재 원소(key)를 앞쪽 원소들과 비교한다.

2. key보다 큰 값은 한 칸씩 뒤로 이동시키며 빈 자리를 만든다.

3. 적절한 위치가 결정되면 eky를 그 자리에 삽입한다.

4. 이 과정을 리스트의 끝까지 반복하여 정렬을 완성한다.

결과 화면

|  |
| --- |
|  |

문제 2. 임의의 자연수 N을 입력 받아서 N개의 랜덤한 데이터를 생성해 반복문과 조건문 만을 이용하여 쉘정렬의 각 단계별 결과를 보여주는 코드를 작성하고 반드시 코드에 대한 자세한 설명을 추가하시오. 초기 쉘정렬의 간격은 N/2로 한다.

입력 예제

생성할 랜덤 변수 개수를 입력하세요: 10

출력 예제

생성할 랜덤 변수 개수를 입력하세요: 10

생성된 list : [92, 21, 5, 100, 31, 64, 47, 78, 77, 12]

쉘 정렬 시작:

간격 5 단계: [64, 21, 5, 77, 12, 92, 47, 78, 100, 31]

간격 2 단계: [5, 21, 12, 31, 47, 77, 64, 78, 100, 92]

간격 1 단계: [5, 12, 21, 31, 47, 64, 77, 78, 92, 100]

최종 정렬 결과: [5, 12, 21, 31, 47, 64, 77, 78, 92, 100]

소스코드 및 설명

|  |
| --- |
| import random  def main():      test = False      if test:          array\_size = 10          array = [92, 21, 5, 100, 31, 64, 47, 78, 77, 12]      else:          array\_size = int(input("생성할 랜덤 변수 개수를 입력하세요: "))          array = random.sample(range(1, 101), array\_size)      ans = []      ans.append("생성된 list : [" + ", ".join(map(str, array)) + "]")      ans.append("쉘 정렬 시작:")      gap = array\_size // 2      while gap > 0:          chunk\_size = array\_size // gap          chunks = []          # gap 크기에 맞춰 chunk 요소 구성          for i in range(gap):              chunk = []              for j in range(chunk\_size):                  idx = i + j \* gap                  if idx < array\_size:                      chunk.append(idx)              chunks.append(chunk)          # 각 chunk에 대한 삽입 정렬 실행          insertion\_sort(array, chunks)          ans.append(f"간격 {gap} 단계: [" + ", ".join(map(str, array)) + "]")          gap = gap // 2      ans.append("최종 정렬 결과: [" + ", ".join(map(str, array)) + "]")      final\_ans = "\n".join(ans)      print(final\_ans)  def insertion\_sort(array, chunks):      for chunk in chunks:          for i in range(1, len(chunk)):              core\_val = array[chunk[i]]              j = i              # core\_val보다 큰 값들을 오른쪽으로 한 칸씩 밀기              while j > 0 and array[chunk[j - 1]] > core\_val:                  array[chunk[j]] = array[chunk[j - 1]]                  j -= 1              array[chunk[j]] = core\_val  if \_\_name\_\_ == "\_\_main\_\_":      main() |

설명 – 쉘 정렬 과정

1. 초깃값으로 gap = 배열 크기 // 2를 설정한다.

2. 각 gap 값에 대해 배열을 여러 개의 부분 리스트(chunk) 로 나눈다. (예: gap=5라면, 인덱스가 (0,5,10,…), (1,6,11,…) … 이런 식으로 묶인다.)

3. 각 부분 리스트마다 삽입 정렬을 수행하여 원소들을 정렬한다.

4. 정렬이 끝나면 gap을 절반으로 줄이고 다시 반복한다.

5. gap이 1이 되면, 사실상 일반 삽입 정렬과 동일하게 동작하며 배열 전체가 완전히 정렬된다.

결과 화면

|  |
| --- |
|  |

문제 3. 임의의 자연수 N을 입력 받아서 N개의 랜덤한 데이터를 생성해 반복문과 조건문 만을 이용하여 퀵정렬의 각 단계별 결과를 보여주는 코드를 작성하고 반드시 코드에 대한 자세한 설명을 추가하시오.

입력 예제

생성할 랜덤 변수 개수를 입력하세요: 10

출력 예제

생성된 list : [8, 26, 85, 84, 45, 29, 81, 36, 40, 9]

퀵 정렬 시작:

생성된 피벗의 값 :8

정렬 단계: [8, 26, 85, 84, 45, 29, 81, 36, 40, 9]

생성된 피벗의 값 :26

정렬 단계: [8, 9, 26, 84, 45, 29, 81, 36, 40, 85]

생성된 피벗의 값 :84

정렬 단계: [8, 9, 26, 40, 45, 29, 81, 36, 84, 85]

생성된 피벗의 값 :40

정렬 단계: [8, 9, 26, 29, 36, 40, 81, 45, 84, 85]

생성된 피벗의 값 :81

정렬 단계: [8, 9, 26, 29, 36, 40, 45, 81, 84, 85]

생성된 피벗의 값 :29

정렬 단계: [8, 9, 26, 29, 36, 40, 45, 81, 84, 85]

최종 정렬 결과: [8, 9, 26, 29, 36, 40, 45, 81, 84, 85]

소스코드 및 설명

|  |
| --- |
| import random  def main():      test = False      if test:          array\_size = 10          array = [8, 26, 85, 84, 45, 29, 81, 36, 40, 9]      else:          array\_size = int(input("생성할 랜덤 변수 개수를 입력하세요: "))          array = random.sample(range(1, 101), array\_size)      ans = []      ans.append("생성된 list : [" + ", ".join(map(str, array)) + "]")      ans.append("퀵 정렬 시작: ")      section = [0, len(array) - 1]      recursive\_quick\_sort(array, section, ans)      ans.append("최종 정렬 결과: [" + ", ".join(map(str, array)) + "]")      final\_ans = "\n".join(ans)      print(final\_ans)  def recursive\_quick\_sort(array, section, ans):      left, right = section      if right - left < 1:          return      pivot = array[left]      ans.append(f"생성된 피벗의 값: {pivot}")      low = left + 1      high = right      while True:          while low <= high and array[low] <= pivot:              low = low + 1          while low <= high and array[high] >= pivot:              high = high - 1          if low > high:              break          array[low], array[high] = array[high], array[low]      array[left], array[high] = array[high], array[left]      low\_section = [left, high - 1]      high\_section = [high + 1, right]      ans.append("정렬 단계: [" + ", ".join(map(str, array)) + "]")      recursive\_quick\_sort(array, low\_section, ans)      recursive\_quick\_sort(array, high\_section, ans)  if \_\_name\_\_ == "\_\_main\_\_":      main() |

설명 – 퀵 정렬 과정

1. 정렬할 구간의 왼쪽 인덱스와 오른쪽 인덱스를 받는다.

2. 원소가 1개 이하라면 종료한다.

설명 - 피벗 선택 및 분할

1. 구간의 첫 번째 원소를 피벗으로 설정한다.

2. low는 왼쪽에서 오른쪽으로 이동하며 pivot보다 큰 원소를 찾고, high는 오른쪽에서 왼쪽으로 이동하며 pivot보다 작은 원소를 찾는다.

3. low와 high이 교차하기 전까지 두 원소를 교환한다.

4. 교차하면 pivot과 high 위치의 값을 교환하여 pivot을 제자리로 옮긴다.

결과 화면

|  |
| --- |
|  |

문제 4. 임의의 자연수 N을 입력 받아서 N개의 랜덤한 데이터를 생성해 반복문과 조건문 만을 이용하여 합병정렬의 각 단계별 결과를 보여주는 코드를 작성하고 반드시 코드에 대한 자세한 설명을 추가하시오.

입력 예제

생성할 랜덤 변수 개수를 입력하세요: 20

출력 예제

생성된 list : [66, 31, 10, 75, 39, 6, 9, 35, 50, 81, 31, 19, 25, 44, 50, 29, 49, 69, 44, 26]

합벙 정렬 시작:

크기 1 단계: [31, 66, 10, 75, 6, 39, 9, 35, 50, 81, 19, 31, 25, 44, 29, 50, 49, 69, 26, 44]

크기 2 단계: [10, 31, 66, 75, 6, 9, 35, 39, 19, 31, 50, 81, 25, 29, 44, 50, 26, 44, 49, 69]

크기 4 단계: [6, 9, 10, 31, 35, 39, 66, 75, 19, 25, 29, 31, 44, 50, 50, 81, 26, 44, 49, 69]

크기 8 단계: [6, 9, 10, 19, 25, 29, 31, 31, 35, 39, 44, 50, 50, 66, 75, 81, 26, 44, 49, 69]

크기 16 단계: [6, 9, 10, 19, 25, 26, 29, 31, 31, 35, 39, 44, 44, 49, 50, 50, 66, 69, 75, 81]

최종 정렬 결과: [6, 9, 10, 19, 25, 26, 29, 31, 31, 35, 39, 44, 44, 49, 50, 50, 66, 69, 75, 81]

소스코드 및 설명

|  |
| --- |
| import random  def main():      test = False      if test:          array\_size = 20          array = [66, 31, 10, 75, 39, 6, 9, 35, 50, 81, 31, 19, 25, 44, 50, 29, 49, 69, 44, 26]      else:          array\_size = int(input("생성할 랜덤 변수 개수를 입력하세요: "))          array = random.sample(range(1, 101), array\_size)      ans = []      ans.append("생성된 list : [" + ", ".join(map(str, array)) + "]")      ans.append("합병 정렬 시작:")      n = len(array)      size = 1  # 처음에는 1개씩 묶어서 비교/합병 시작      # 크기를 1, 2, 4, 8... 늘려가며 병합 수행      while size < n:          # i는 병합할 구간의 시작점          for i in range(0, n, size \* 2):              mid = i + size              end = min(i + size \* 2, n)              if mid < end:  # 실제로 병합할 오른쪽 부분이 존재할 때만                  merged = merge(array[i:mid], array[mid:end])                  array[i:end] = merged          ans.append(f"크기 {size} 단계: [" + ", ".join(map(str, array)) + "]")          size \*= 2      ans.append("최종 정렬 결과: [" + ", ".join(map(str, array)) + "]")      final\_ans = "\n".join(ans)      print(final\_ans)  def merge(left, right):      """두 개의 정렬된 배열을 병합하는 함수"""      result = []      l\_ptr = r\_ptr = 0      # 두 배열에서 작은 값을 하나씩 꺼내 result에 채움      while l\_ptr < len(left) and r\_ptr < len(right):          if left[l\_ptr] <= right[r\_ptr]:              result.append(left[l\_ptr])              l\_ptr += 1          else:              result.append(right[r\_ptr])              r\_ptr += 1      # 남아 있는 원소들 붙여주기      result.extend(left[l\_ptr:])      result.extend(right[r\_ptr:])      return result  if \_\_name\_\_ == "\_\_main\_\_":      main() |

설명 – 합병 정렬 시작

1. 정렬할 리스트의 크기를 1로 시작한다.

2. 크기는 매 단계마다 2배씩 늘어나며, 리스트를 size 길이의 그룹으로 나눈 뒤 인접한 두 그룹을 병합한다.

설명 – 병합 과정

1. 두 개의 정렬된 부분 배열을 입력받아 하나의 정렬된 배열로 합친다.

2. 각 배열의 포인터를 사용해 작은 값을 하나씩 꺼내 결과에 추가한다.

3. 둘 중 하나의 배열이 먼저 소진되면, 남은 원소들을 그대로 붙인다.

결과 화면

|  |
| --- |
|  |