

**자료 구조 과제 4**

**과목명 자료구조**

**담당교수 김희철교수님**

**제출일 20210507**

**전공 컴퓨터전자시스템**

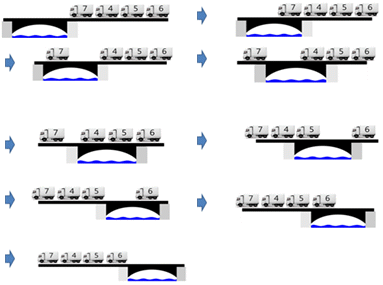
**학번 201904458**

**이름 이준용**

* **문제 1-2 번**

4-2) 강을 가로지르는 하나의 차선으로 된 다리가 하나 있다. 이 다리를 n개의 트럭이 건너가려고 한다. 트럭의 순서는 바꿀 수 없으며, 트럭의 무게는 서로 같지 않을 수 있다. 다리 위에는 단지 w 대의 트럭만 동시에 올라갈 수 있다. 다리의 길이는 w 단위길이(단위길이 = 트럭 길이)이며, 각 트럭은 하나의 단위시간에 하나의 단위길이 만큼만 이동할 수 있다고 가정한다. 동시에 다리 위에 올라가 있는 트럭들의 무게의 합은 다리의 최대하중인 L보다 작거나 같아야 한다. 참고로, 다리 위에 완전히 올라가지 못한 트럭의 무게는 다리 위의 트럭들의 무게의 합을 계산할 때 포함하지 않는다고 가정한다.

예를 들어, 다리의 길이 w는 2, 다리의 최대하중 L은 10, 다리를 건너려는 트럭이 트럭의 무게가 [7, 4, 5, 6]인 순서대로 다리를 오른쪽에서 왼쪽으로 건넌다고 하자. 이 경우 모든 트럭이 다리를 건너는 최단시간은 아래의 그림에서 보는 것과 같이 8이다.



다리의 길이와 다리의 최대하중, 그리고 다리를 건너려는 트럭들의 무게가 순서대로 주어졌을 때, 모든 트럭이 다리를 건너는 최단시간을 구하는 프로그램을 작성하시오.

**요구조건: 큐를 이용해야 함**

**입력**

첫 번째 줄에는 세 개의 정수 n(1≤n≤1000), w(1≤w≤100), L(10≤L≤1000)이 주어지는데, n은 다리를 건너는 트럭의 수, w는 다리의 길이, 그리고 L은 다리의 최대하중을 나타낸다. 입력의 두 번째 줄에는 n개의 정수 a1, a2, ..., an이 주어지는데, ai는 i번째 트럭의 무게를 나타낸다.

**출력**

모든 트럭들이 다리를 건너는 최단시간을 출력하라.

입출력 예

|  |  |
| --- | --- |
| 입력 예제 1 | 출력 예제 1 |
| 4 2 10  7 4 5 6 | 8 |

|  |  |
| --- | --- |
| 입력 예제 2 | 출력 예제 2 |
| 1 100 100  10 | 101 |

|  |  |
| --- | --- |
| 입력 예제 3 | 출력 예제 3 |
| 10 100 100  10 10 10 10 10 10 10 10 10 10 | 110 |

* **알고리즘 및 자료구조**

배열 큐가 아닌 LinkedQueue로 구현했습니다.

클래스 Node

클래스 Queue

-isEmpty(self)

-enqueue(self, e)

-dequeue(self)

-peek(self)

Answer 변수에 세 개의 정수 n(1≤n≤1000), w(1≤w≤100), L(10≤L≤1000)이 주어지는데, n은 다리를 건너는 트럭의 수, w는 다리의 길이, 그리고 L은 다리의 최대하중을 받고,

Tmp 변수에 두번째 줄에 입력 되는 n개의 정수 i번째 트럭의 무게를 받습니다.

Truck과 bridge변수는 queue클래스에 연결

Time = 최단시간

Bridge\_sum = 최대 하중

큐에 지나갈 수 있는 트럭을 넣고 하중을 초과하지 않으면 그 다음 트럭을 계속 넣습니다.

초과한다면 0을 enqueue합니다.

큐 사이즈가 최대하중을 초과하면 큐에 있는 원소들을 하나씩 dequeue하여 트럭의 무게 또는 0값을 제거합니다.

트럭이 dequeue 될 때마다 time을 1씩 증가하고 마지막 트럭이 dequeue되면 time 출력

* **느낀 점**

처음에는 문제를 해석하고 이해하는데 시간이 많이 걸렸던 것 같습니다. 브릿지와 트럭의 무게를 enqueue하는 과정에서 하나씩 해보니 이해할 수 있게 되었습니다.

이런 문제들을 큐라는 자료구조를 접목시켜 풀 수 있다는 것에 큐에 대해 많이 알아 가는 시간이 었습니다.

* **프로그램 코드**
* # 1.제목:한국외대 자료구조 과제 4(4-1-2)  
  # 2.날짜:20210501  
  # 다리의 길이와 다리의 최대하중, 그리고 다리를 건너려는 트럭들의 무게가 순서대로 주어졌을 때,  
  # 모든 트럭이 다리를 건너는 최단시간을 구하는 프로그램을 작성하시오.  
    
  # \*\*\*\* 배열 큐가 아닌 LinkedQueue로 구했습니다. \*\*\*\*  
  *'''  
  4 2 10  
  7 4 5 6  
  '''*'''  
  8  
  '''  
  class Node:  
   def \_\_init\_\_(self, element):  
   self.data = element  
   self.link = None  
  class Queue:  
   def \_\_init\_\_(self):  
   self.front = self.rear = None # data 받기 전에는 front, rear는 None 상태  
    
   def isEmpty(self):  
   return self.front == None # front가 None일 경우 data 아무것도 없음  
    
   def enqueue(self, e):  
   newNode = Node(e)  
   if self.front == None:  
   self.front = self.rear = newNode  
   else:  
   self.rear.link = newNode  
   self.rear = newNode  
    
   def dequeue(self):  
   if self.isEmpty():  
   print("Queue empty")  
   return 0  
   e = self.front.data  
   self.front = self.front.link  
   if self.front == None:  
   self.rear = None  
   return e  
    
   def peek(self):  
   if self.isEmpty():  
   print("Queue empty")  
   else:  
   return self.front.data # self.front 만 리턴하면 type이 node가 됨  
   # 그러므로 self.front.data로 element를 뽑아냄  
  answer = input().split(" ") # n == 다리를 건너는 트럭의 수 w == 다리의 길이 L == 다리의 최대하중  
   # 세 개의 정수 n(1≤n≤1000), w(1≤w≤100), L(10≤L≤1000)  
  tmp = input().split(" ") # 입력의 두 번째 줄에는 n개의 정수 a1, a2, ..., an이 주어지는데,  
   # ai는 i번째 트럭의 무게를 나타낸다.  
    
  truck = Queue() # truck변수와 bridge변수 Queue 클래스에 연결  
  bridge = Queue()  
  time = 0 # 최단시간  
  bridge\_sum = 0 # 최대 하중  
    
  for i in tmp:  
   truck.enqueue(int(i))  
  for i in range(int(answer[1])):  
   bridge.enqueue(0) # 다리무게가 0 인 트럭을 세워놓고 enqueue하면 트럭의 무게를 할당해주는 방식으로 풀어냈습니다.  
    
  while not bridge.isEmpty():  
   if truck.isEmpty(): # bridge를 할당했지만 truck이 비어 있을 경우  
   bridge.dequeue() # bridge를 dequeue함.  
   else: # bridge 할당하고 truck무게도 할당했을 경우  
   bridge\_sum -= bridge.dequeue() # 최대 하중 = 최대하중 - 브릿지 하중  
    
   if bridge\_sum + truck.peek() <= int(answer[2]): # 최대하중 + 맨앞 트럭 무게가 그다음 무게의 트럭보다 작을 경우  
   tmp = truck.dequeue() # 그다음 트럭 dequeue  
   bridge\_sum += tmp # 브릿지 최대하중에 건너는 트럭 무게 더함  
   bridge.enqueue(tmp) # 트럭이 건너면 브릿지 무게 enqueue  
   else:  
   bridge.enqueue(0) # 브릿지 최대하중 못버팀  
   time += 1 # 트럭 건널때마다 time++  
    
  print(time)
* **문제 1-3번**

4-3) (간단한 연결리스트 만들기 연습)

교과목 수강자 관리를 위한 다음 명령어를 처리하는 프로그램을 작성하시오. 수강자 정보는 학번(문자열)이다.

**명령어 양식**

N id // 학번이 id인 학생이 수강신청을 함

**C id // 학번이 id인 학생이 수강신청을 취소함**

S // 수강 학생수를 출력

P // 수강 학생들을 학번순서(오름차순)대로 출력

Q // 끝내기

**명령어 입력에 있어서 오류는 없다고 가정한다.** 예를 들어, id가 1111인 학생이 수강신청을 하였을 경우, N 1111의 명령어는 주어지지 않는다. id가 1112인 학생이 수강신청을 하지 않았을 경우, C 1112인 명령어는 주어지지 않는다.

**제약조건: 수강자리스트를 연결리스트로 구현함 (수강자 노드들은 임의의 순서로 연결할 수 있음)**

수강자리스트 연산은 다음을 포함하여야 함

insert(st\_id) # 리스트에 수강자\_id (문자열)를 삽입하는 함수

delete(st\_id) # 리스트에서 수강자\_id (문자열)를 삭제하는 함수

size() # 리스트 원소(수강자) 개수 반환

print() # 리스트의 원소(수강자)들을 오름차순으로 출력하는 함수

class stList:

def \_\_init\_\_(self);

self.head = None

def insert(self, st\_id);

def delete(self, st\_id);

def size(self);

def print(self)

...

**main 프로그램 예**

L = stList()

while True:

command = input().split()

if command[0] == 'N':

# st\_id(즉, command[1])를 수강자리스트에 insert: L.insert(command[1])

elif command == 'C':

# st\_id(즉, command[1])를 수강자리스트에서 삭제: L.delete(command[1])

elif command == 'S':

# 수강자리스트의 원소 수를 출력: L.size() 출력

elif command == 'P':

# 수강자리스트의 원소들을 오름차순으로 출력: L.print()

else if command == 'Q':

break;

입출력 예:

|  |  |
| --- | --- |
| 입력 예 1 | 출력 예 1 |
| N 1113  N 1112  N 1111  N 1115  S  C 1112  P  S  N 1112  P  Q | 4 # S  1111 1113 1115 # P  3 # S  1111 1112 1113 1115 # P |

* **알고리즘 및 자료구조**

클래스 node

클래스 StList

-isempty(self)

-insert(self, st\_id) -> 교수님께 피드백 받은 대로 노드 삽입하는 부분에서 새로운 data를 받을 때 마다 head 다음 첫 노드로 연결합니다. 노드 끝에 연결하는 것은 주석 처리했습니다.

-delete(self, st\_id) -> 노드에서 노드가 하나만 있을 때 삭제하는 경우와 저장된 노드에서 중간 값을 지우는 경우로 나눠서 작성했습니다.

-sizelist(self) -> list의 현재 크기를 반환하는 함수입니다. 명령어 S에서 노드 개수 출력에 이용됩니다.

-print(self) -> node의 data를 배열에 head의 link가 None이 될 때까지 배열에 append 합니다.

sorted(arr, key=str)로 문자열로 정렬하는 key=str 사용했습니다.

그리고 정렬된 배열 반환합니다.

* **느낀 점**

연결 리스트를 사용하여 학생 수강신청 프로그램의 여러 함수를 작성해보면서 LinkedList의 ADT들의 함수들을 이해할 수 있었습니다. 교수님께 질문 드렸던 delete 부분과 다른 학우가 질문했던 8번 케이스인 ex) 111-113 처리하는 부분을 고쳤습니다. 감사합니다.

* **프로그램 코드**
* # 1.제목:한국외대 자료구조 과제 4(4-1-3)  
  # 2.날짜:20210504  
  # 4-3) (간단한 연결리스트 만들기 연습)  
  class Node:  
   def \_\_init\_\_(self, element):  
   self.data = element  
   self.link = None  
    
  class StList:  
   def \_\_init\_\_(self):  
   self.head = None  
    
   def isempty(self): # 노드의 개수를 반환 해주는 함수에서 반환 값이 0인 경우   
   if self.sizelist() != 0:   
   return False # 비어있지 않음 -> False  
   else:   
   return True # 비어있음 ->True  
    
   def insert(self, st\_id): # 교수님께 피드백 받은 대로 노드 삽입하는  
   # 부분에서 새로운 data를 받을 때 마다 head 다음 첫 노드로 연결합니다.  
   newnode = Node(st\_id)  
   newnode.link = self.head  
   self.head = newnode  
   # newnode = Node(st\_id) # 새로받은 data를 노드 끝에 연결하는 것은 주석 처리했습니다.  
   # if self.head is None:  
   # self.head = newnode  
   # else:  
   # cur = self.head  
   # while cur.link is not None:  
   # cur = cur.link  
   # cur.link = newnode  
    
   def delete(self, st\_id):  
   # 지우는 값이 맨 앞 부분 일 때  
   if self.head.data == st\_id:  
   tmp = self.head  
   self.head = self.head.link  
   del tmp  
   else:  
   # 지우는 값이 중간값일때  
   node = self.head  
   while node.link:  
   if node.link.data == st\_id:  
   temp = node.link  
   node.link = node.link.link  
   del temp  
   return  
   else:  
   node = node.link  
    
   def sizelist(self) -> int: # list 크기 반환  
   count = 0  
   current = self.head  
   while current is not None:  
   count += 1  
   current = current.link  
   return count  
    
   def print(self):  
   arr1 = []  
   newfound = self.head  
   while newfound:  
   if newfound.link is not None:  
   arr1.append(newfound.data)  
   newfound = newfound.link  
   else:  
   arr1.append(newfound.data)  
   newfound = newfound.link  
   int\_li = sorted(arr1, key=str) # 맨 앞자리 수 기준 으로 정렬 key =str 이용  
   # int\_li = sorted(map(str, arr1))  
   return int\_li  
    
    
  L = StList()  
  while True:  
   command = input().split()  
   if command[0] == 'N': # st\_id(즉, command[1])를 수강자리스트에 insert: L.insert(command[1])  
   L.insert(command[1])  
   elif command[0] == 'C': # st\_id(즉, command[1])를 수강자리스트에서 삭제: L.delete(command[1])  
   L.delete(command[1])  
   elif command[0] == 'S': # 수강자리스트의 원소 수를 출력: L.size() 출력  
   print(L.sizelist()) # L.size하면 int형을 반환 할 수 없다고 에러나서 sizelist로 이름 변경했습니다.  
   elif command[0] == 'P': # 수강자리스트의 원소들을 오름차순으로 출력: L.print()  
   arr = L.print() # L.print()  
   for i in arr:  
   print(i, end=' ')  
   elif command[0] == 'Q': # 종료  
   break  
  # 교수님께 질문 드렸던 delete 부분과 다른 학우가 질문했던 8번 케이스인 ex) 111-113 처리하는 부분을 고쳤습니다.