**자료구조 2번째 실습 보고서**

202411292 박준건

* 1. **Insertion**

링크드 리스트와 배열 모두 O(1)만큼의 시간 복잡도를 가지는 삽입 연산을 수행하도록 하였습니다.(링크드 리스트는 헤더에 추가, 배열은 맨 뒷자리에 추가)

배열의 경우엔 맨 뒤 인덱스에 접근 후 대입하는 방법을 사용했고, 링크드 리스트는 새 리스트를 생성한 다음에 이중 포인터로 받은 Header의 주소를 새 포인터로 수정, 이후 새 포인터의 next에 원래 header의 주소를 넣는 방식을 사용했습니다.

텍스트, 스크린샷, 폰트, 시계이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

둘 다 유의미한 시간의 차이가 생기진 않았습니다만. 아무래도 링크드 리스트는 배열에 비해 약 3번 정도의 과정을 더 거치므로 실행 시간이 늘어났다고 추정합니다.

* 1. **Random access for read:**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

베열의 경우엔 인덱스로 바로 접근하는 방법을, 링크드 리스트의 경우엔 부여받은 인덱스 번호만큼 타고 가서 읽어오는 방식을 수행하게 됩니다. 전자는 O(1), 후자는 O(n)의 시간이 걸리게 됩니다. 이 때문에 배열의 엑세스 시간이 링크드 리스트의 랜덤 엑세스 시간보다 현저히 더 낮게 나오는 것을 확인할 수 있었습니다.

* 1. **Random access for deletion:**

텍스트, 스크린샷, 폰트, 회로이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

랜덤 삭제의 경우 링크드 리스트와 배열 모두 많은 시간을 필요로 합니다.

배열의 경우 index의 요소를 삭제한 뒤 index<n인 모든 n번째 요소를 전부 n-1번째로 당겨오는 식의 방법으로 시프트를 수행하였고,

링크드 리스트는 삭제하려는 노드의 전 노드와 후 노드를 이어버리고 삭제하려는 노드를 free() 안에 넣는 식으로 삭제를 수행했습니다.

접근 과정에서 배열은 인덱스로 바로 접근할 수 있지만, 링크드 리스트에는 접근에 또 다른 시간이 필요했습니다. 따라서 배열에 비해 더 긴 시간이 걸리는 것을 확인할 수 있었습니다. 시간 복잡도는 둘 모두 O(n)입니다.

**2-1 time complexity측정.**

1. **Insert operation의 1회 수행시간**

다음의 경우 링크드 리스트는 첫 번째(헤드)에, 배열은 마지막 인덱스에 계속 추가해나가는 방법을 사용했습니다. 이 경우 두 자료형 모두 시간 복잡도는 O(1)입니다. N이 늘어난다고 해서 작업 과정이 더 길어지지 않고, 일정한 작업을 수행하기만 하면 되기 때문입니다.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**삭제의 경우에는 다음과 같습니다**

텍스트, 번호, 소프트웨어, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

삭제의 경우엔 링크드 리스트와 배열 모두 O(n)만큼의 수행 시간을 필요로 합니다. 하지만 1회의 삭제 시간을 비교하는 것이니만큼, 실제 실행 시간은 매우 작은 단위로 나타내졌습니다. 다만 그 중에도 요소의 개수가 늘어날수록 실행 시간이 점점 늘어나는 현상을 관찰할 수 있었습니다.

**Read 함수의 실험 방법 구상**

Read 오퍼레이션의 경우엔 배열은 모든 경우에 O(1)인 반면, 링크드리스트는 O(n)의 모습을 보일 것입니다. 따라서, 앞서 구상한 실험 방식과 동일하게 n을 점차 늘려가며 배열과 링크드리스트 사이의 시간 차이를 측정하는 방식을 이용할 것입니다. 이 경우 배열과 링크드 리스트 사이의 차이점을 확인하는 실험이 될 것으로 사료됩니다.