**자료구조,** 우현제**,** 2022 1st semester

**자료구조를 활용한 증권 거래 시스템 구현**

**Final Report**

201823869

E-Business

조성우



**INDEX**

**1. 개요**

**2. 시스템 핵심 기능**

**3. 활용한 자료구조**

**4. 테스트 결과**

**5. 한계점**

1. **개요**

중간 계획서에서 언급했듯, 평소 제가 일상에서 자주 접할 수 있었던 주요 IT 서비스 중 CRUD 기능의 속도와 효율성이 중요 시 되는 시스템을 주제로 선택하여, 적합한 자료구조와 알고리즘 선택의 중요성에 몰입하여 과제를 진행해보고 싶었습니다. 따라서 특히나 모든 기능의 속도와 효율이 중요시되는 증권거래시스템을 선택하였습니다.

또한 증권 거래 시스템에서 매수주문과 매도주문 간 호가가 겹칠 경우 즉시 주문간 거래를 체결시켜야 하는데, 주문 체결의 속도를 높이기 위해 어떤 자료구조를 사용할 수 있을지 몰입해보며, 지금까지 배웠던 자료구조들을 능동적으로 복습/학습할 수 있다고 생각하여 증권 거래 시스템이 좋은 주제가 될 것이라고 생각하였습니다.

1. **2차 수정 및 보완점에 대한 설명**

1차 보고서에서는 시간과 개인의 역량 관계상 ‘특정 주문 취소 기능’에 더해 트레이딩 시스템의 가장 핵심이라고 할 수 있는 ‘거래간 주문 체결 기능’을 구현하지 못하였습니다. 이번 최종 프로그램에서는 **‘시스템 상 등록된 거래간 주문 체결 기능’을 추가적으로 구현하였습니다**. 하지만 아쉽게도 추가적으로 지적해 주셨던 특정 주문 취소 기능은 아직 구현하지 못하였습니다.

1. **활용한 자료구조**

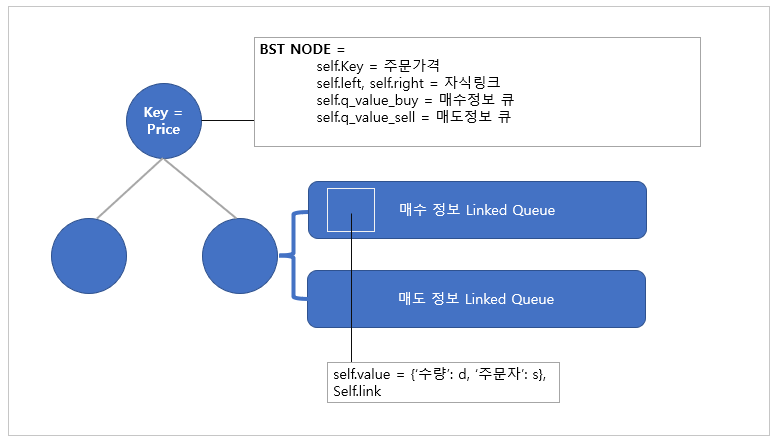
**자료구조에 대한 고민**

증권 거래 시스템은 탐색과 삽입/삭제 연산의 모든 연산속도가 모두 효율적으로 보장되어야 하는 특성, 매수주문과 매도주문의 빠른 체결을 위해 정렬이 유지되어야 한다는 특성이 요구되므로 기존에는 MAX값과 MIN값을 빠르게 찾아내서 거래를 체결시키기 위해 힙을 이용하는 것이 효율적이겠다 생각하였고 힙을 사용한 구현을 계획했습니다.

하지만 느슨한 정렬이라는 불완전성으로 호가창을 보여줄 때 정렬의 효율성이 떨어질 수 있다고 판단하였고 다른 자료구조를 고민하게 되었습니다. 그 결과 최근에 배운 **이진탐색트리를** 사용하여 구현해보자 생각했습니다.

하지만 BST를 도입하기 전 또 하나의 고민이 생겼습니다. 가격을 key값으로 사용한다면 반드시 중복을 발생시키는데, BST에서 중복이 발생하면 BST 사용의 가치가 떨어지기 때문에 BST의 NODE에 큐(매수큐/매도큐)를 붙여 중복 키에 대한 값을 해당 노드 내의 큐에 저장하는 자료구조를 구현함으로써 이를 극복하고자 했습니다.

그 모습은 아래와 같습니다.



노드 클래스 내에 매수 큐와 매도 큐를 두었습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<핵심 인자>

- **isTypeBuy : bool** 파라미터는 들어오는 주문 데이터가 매수주문인지 매도주문인지를 담은 값 입니다. True라면 값을 매수주문 큐에, False라면 매도주문 큐에 삽입합니다.

- **key :** 해당 BST의 키값은 ‘가격’입니다. 가격을 사용하여 BST의 정렬을 유지시켜야 하기 때문에 가격을 골랐고 큐를 사용하여 중복 KEY를 허용했습니다.

**왜 큐인가? :** 증권 거래 시스템 상 거래가 체결될 때, 같은 가격의 주문이 존재할 경우 먼저 들어온 주문을 먼저 체결시키기 때문에 선입선출이 요구되기 때문에, 가격이 중복된 값의 나열에서 큐를 사용하였습니다.

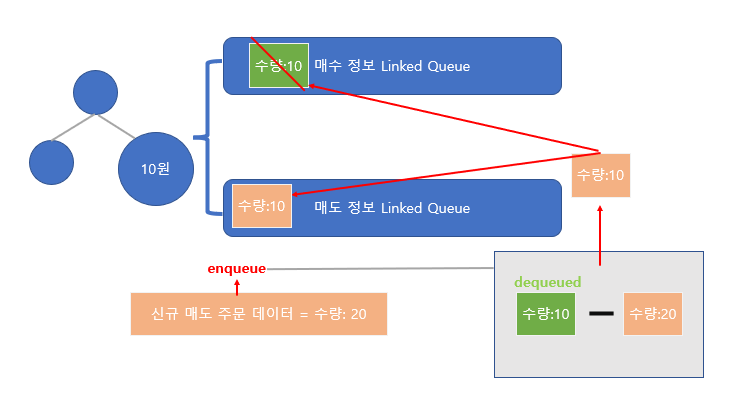
**왜 매수큐 매도큐를 따로두었는가? :** 매수 큐와 매도 큐를 구분함으로써, 양 큐 간의 값이 동시에 존재할 경우 동일가격의 반대매매가 동시에 존재하는 것이 되므로 중복 가격에 대한 반대매매 삽입 연산 발생 시 매수큐와 매도큐 내의 값 삭제 또한 비교를 통해 동시적으로 처리할 수 있다는 생각을 하였습니다. 하지만 아쉽게도 시간관계상 주문 체결 기능은 구현하지 못했습니다.

1. **시스템 핵심 기능**
2. 정렬 특성: BST를 활용하여 항상 정렬된 모습으로 데이터를 유지 및 출력할 수 있습니다.
3. 주문 데이터 삽입
   1. **일반 삽입:**

동일 가격 BST NODE 내에 반대매매가 존재하지 않는 경우 조건문 하에서 매수 혹은 매도 큐에 일반적으로 삽입합니다.

* 1. **주문 체결 후 삽입:**

삽입 연산 수행 시, 신규 삽입 주문 노드의 가격이 속할 노드가 가진 매수/매도 주문정보 큐를 조회하여 동일가격 노드의 반대매매 큐에 반대주문이 존재했을 시 신규노드와 수량 차이를 조회하고 차이분 만큼을 제외한, 즉, 겹치는 반대 주문의 거래를 성사시킵니다. 그리고 체결되지 않은 남은 주문을 해당하는 매수 or 매도 큐에 삽입합니다.



1. 주문 데이터 탐색 (Max, Min):

최고가/최소가 주문을 탐색하여 반환합니다.

1. 주문 데이터 출력

* 시스템 상 전체 주문 데이터 출력
* 매수 주문 전체 출력
* 매도 주문 전체 출력

1. **테스트 결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

1. **한계점**

1차 보고서의 피드백으로 받았던 큐 내의 값 기반 탐색을 제대로 구현하는 것에 어려움을 겪어 결국 최종 보고서에는 포함시키지 못했습니다. 이번 자료구조 강의 이후로도 해당 기능 구현에 더해 성능 면에 있어서도 지속적으로 해당 프로그램의 한계점을 고민하고 이를 발전시키고자 다짐합니다. 좋은 기회를 마련해 주셔서 감사합니다!