7주차 3차시 알고리즘의 정리

【학습목표】

1. 정렬과 탐색을 정리 한다.
2. 스트링 매칭과 기하 알고리즘을 정리한다.

학습내용1 : 정렬과 탐색

* 정렬 알고리즘 비교표

정렬방식	알고리즘	실행시간	안정적	제자리	특성
비교기반	선택	$O(n^2)$	O/X	O	입력 자료의 순서에 민감하지 않음
	버블	$O(n^2)$	O	O	최악: 키가 역순으로 정렬된 경우
	삽입	$O(n^2)$	O	O	최악: 키가 역순으로 이미 정렬된 경우 제 순서로 거의 정렬되어 있는 상태에서는 꽤 효율적 O(n)
	쉘	$O(n^{1.5})$	X	O	삽입 정렬의 단점 보완 부분배열의 크기와 개수 변화를 위해 순열 사용
	퀵	$O(n^2)$, O(nlogn)	X	O	Partition() - O(n), 분할정복 최악: 정렬되어 있는 경우 최선: 분할원소를 중심으로 정확히 두 부분배열로 나뉠 때
	합병	O(nlogn)	O	X	Merge, 분할정복
	히프	O(nlogn)	X	O	초기히프-O(n), 히프조정-O(logn) 히프 구축 과정 (상향식, 하향식)
데이터 분포	계수	O(n)	O	X	키 값의 (작은) 범위를 아는 경우
	기수	O(n)	O	X	키의 각 자릿수에 대하여 낮은 자리에서 높은 자리의 순으로 정렬.(낮은 자리 먼저)
	버킷	O(n)	O	X	키 값의 범위, 구간 내의 균등한 분포 가정

* 탐색

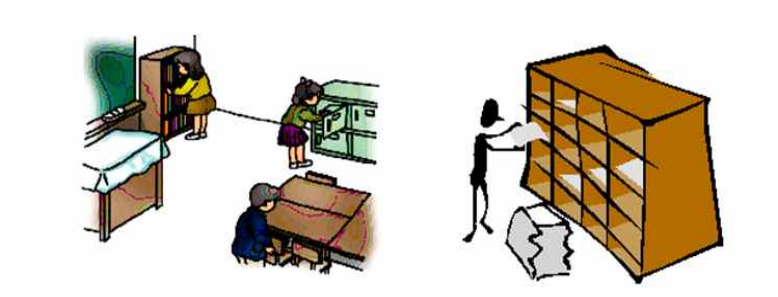
여러 레코드에서 주어진 키를 갖는 레코드를 찾아내는 것을 의미한다.

탐색방법	실행시간	특징	
순차탐색	O(n)	비정렬 데이터에 효과적	
이진탐색	O(logn)	정렬된 레코드의 선형 리스트에 분할 정복을 적용. 삽입이나 삭제가 빈번한 경우 좋지 않다	
이진탐색나무	O(logn), O(n)	연산과정(탐색, 삽입, 삭제), 최악: 경사 나무	
2-3-4 나무	O(logn)	균형나무(탐색나무의 좌우가 거의 같은 높이를 유지)	2,3,4-노드 (링크 k, 키 k-1) 노드의 분할 연산
흑적나무	O(logn)		적링크 사용(3-노드, 4-노드에 속하는 노드들을 연결) 흑적나무 성질(2가지) 회전과 분할 연산
해싱	해시함수(제산 함수, 폴딩 함수, 중간제곱 함수, 비트추출 함수, 숫자 분석 방법) 충돌해결방법(선형조사법, 체이닝)		

1. 해싱(hashing)

⦿ 대부분의 탐색 방법들은 키 값 비교로써 탐색하고자 하는 항목에 접근
- 해싱은 키 값에 대한 산술적 연산에 의해 테이블의 주소를 계산하여 항목에 접근

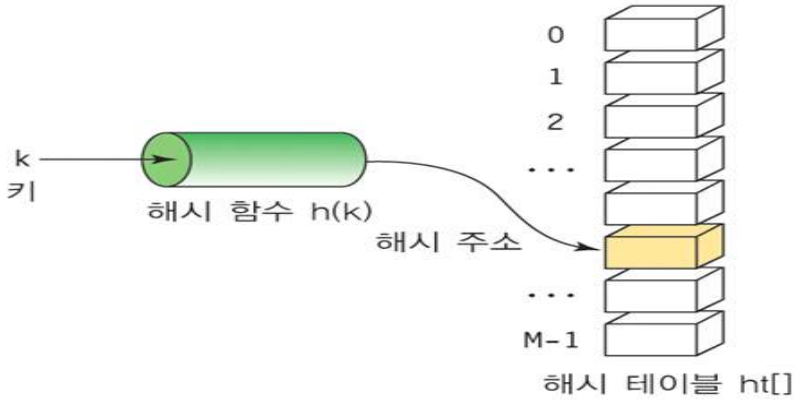
⦿ 해시 테이블(hash table)
- 키 값의 연산에 의해 직접 접근이 가능한 구조로 물건을 정리하는 것과 같다.



⦿ 해시 함수(hash function)

☞ 탐색키를 입력받아 해시 주소(hash address) 생성

☞ 이 해시 주소가 배열로 구현된 해시 테이블(hash table)의 인덱스



⦿ 해시테이블 ht

☞ M개의 버켓(bucket)으로 구성된 테이블

☞ ht[0], ht[1], ...,ht[M-1]의 원소를 가짐

☞ 하나의 버켓에 s개의 슬롯(slot) 가능

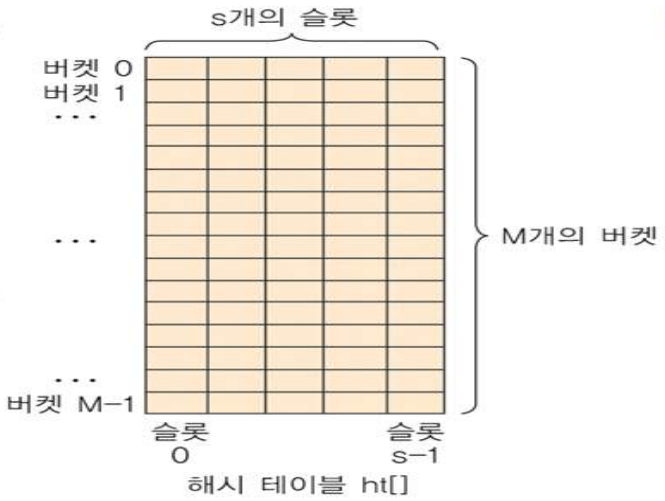
⦿ **충돌(collision)**

☞ 서로 다른 두 개의 탐색키 k1과 k2에 대하여 h(k1) = h(k2)인 경우

⦿ 오버플로우(overflow)

☞ 충돌이 버켓에 할당된 슬롯 수보다 많이 발생하는 것

☞ 오버플로우 해결 방법 반드시 필요



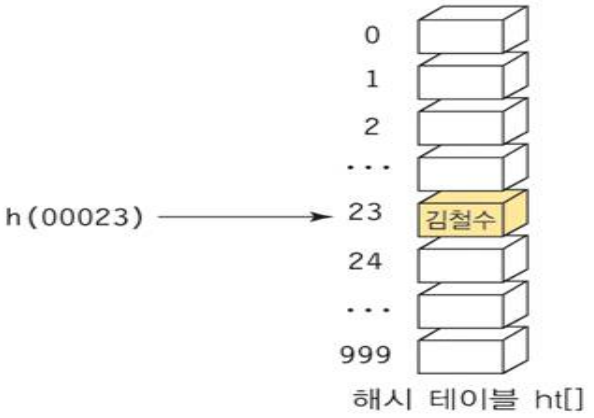
⦿ 학생 정보를 해싱으로 저장, 탐색해보자

☞ 5자리 학번 중에 앞 2자리가 학과 번호, 뒤 3자리가 각 학과의 학생 번호

☞ 같은 학과 학생들만 가정하면 뒤의 3자리만 사용해서 탐색 가능

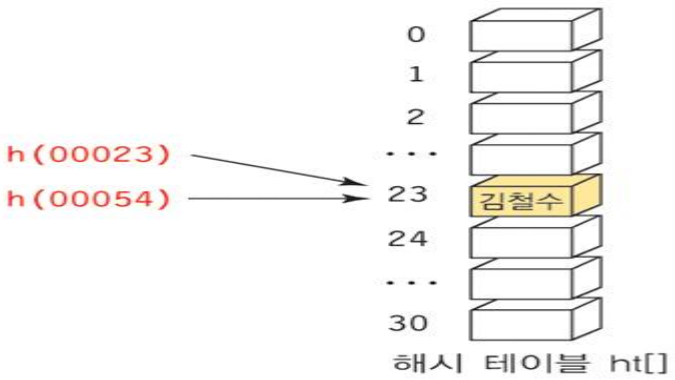
☞ 학번이 00023이라면 이 학생의 인적사항은 해시테이블 ht[23]에 저장

☞ 만약 해시테이블이 1000개의 공간을 가지고 있다면 탐색 시간이 O(1)이 되므로 이상적임



⦿ 실제로는 해시테이블의 크기가 제한되므로, 존재 가능한 모든 키에 대해 저장 공간을 할당할 수 없음

⦿ h(k)= k mod M 의 예에서 보듯이 필연적으로 충돌과 오버플로우 발생함

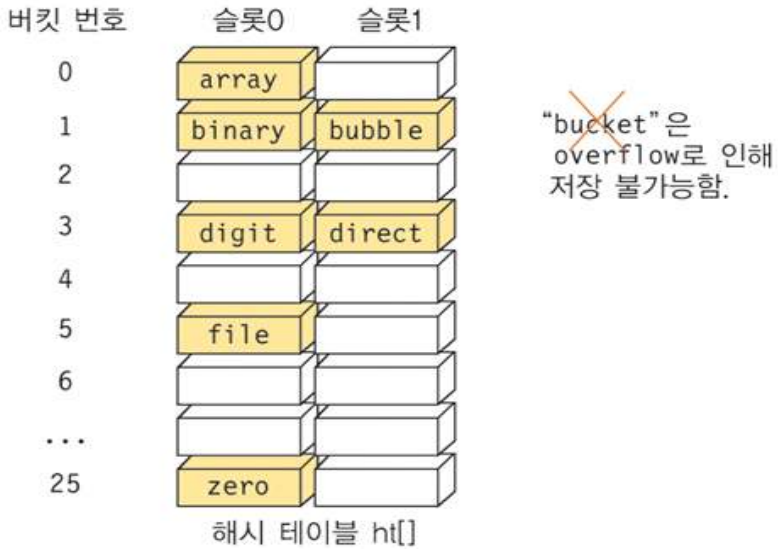


⦿ 알파벳 문자열 키의 해시함수가 키의 첫 번째 문자의 순서라고 하자

h("array")=1

h("binary")=2

⦿ 입력데이터: array, binary, bubble, file, digit, direct, zero, bucket

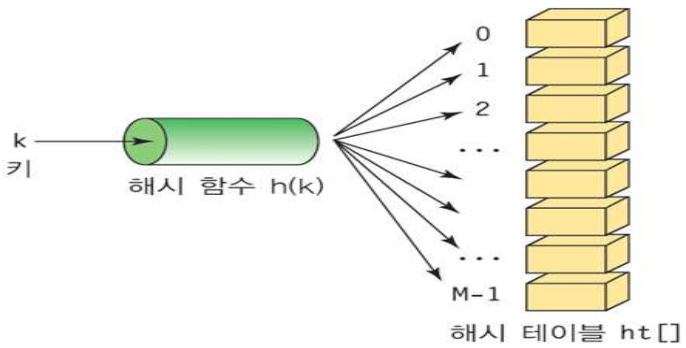


⦿ 좋은 해시 함수의 조건

☞ 충돌이 적어야 한다

☞ 해시함수 값이 해시테이블의 주소 영역 내에서 고르게 분포되어야 한다

☞ 계산이 빨라야 한다



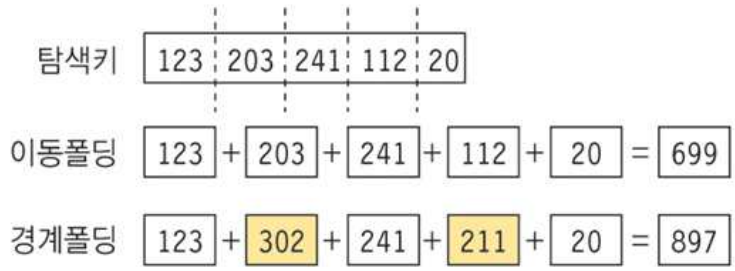
⦿ 제산 함수

☞ h(k)=k mod M

☞ 해시 테이블의 크기 M은 소수(prime number) 선택

⦿ 폴딩 함수

☞ 이동 폴딩(shift folding)과 경계 폴딩(boundary folding)



⦿ 중간제곱 함수

☞ 탐색키를 제곱한 다음, 중간의 몇 비트를 취해서 해시 주소 생성.

⦿ 비트추출 함수

☞ 탐색키를 이진수로 간주하여 임의의 위치의 k개의 비트를 해시 주소로 사용

⦿ 숫자 분석 방법

☞ 키 중에서 편중되지 않는 수들을 해시테이블의 크기에 적합하게 조합하여 사용.

2. 충돌(collision)

서로 다른 탐색 키를 갖는 항목들이 같은 해시 주소를 가지는 현상

충돌이 발생하면 해시 테이블에 항목 저장 불가능

충돌을 효과적으로 해결하는 방법 반드시 필요

해시테이블

* 충돌해결책

선형조사법: 충돌이 일어난 항목을 해시 테이블의 다른 위치에 저장

체이닝: 각 버켓에 삽입과 삭제가 용이한 연결 리스트 할당

⦿ 충돌이 ht[k]에서 발생했다면,

☞ ht[k+1]이 비어 있는지 조사

☞ 만약 비어있지 않다면 ht[k+2] 조사

☞ 비어있는 공간이 나올 때까지 계속 조사

☞ 테이블의 끝에 도달하게 되면 다시 테이블의 처음부터 조사

☞ 조사를 시작했던 곳으로 다시 되돌아오게 되면 테이블이 가득 찬것임

☞ 조사되는 위치: h(k), h(k)+1, h(k)+2,…

⦿ 군집화(clustering)과 결합(Coalescing) 문제 발생

3. 선형조사법

(예) h(k)=k mod 7

1단계 (8) : h(8) = 8 mod 7 = 1(저장)

2단계 (1) : h(1) = 1 mod 7 = 1(충돌발생)

(h(1)+1) mod 7 = 2(저장)

3단계 (9) : h(9) = 9 mod 7 = 2(충돌발생)

(h(9)+1) mod 7 = 3(저장)

4단계 (6) : h(6) = 6 mod 7 = 6(저장)

5단계 (13) : h(13) = 13 mod 7 = 6(충돌 발생)

(h(13)+1) mod 7 = 0(저장)

	1단계	2단계	3단계	4단계	5단계
[0]					13
[1]	8	8	8	8	8
[2]		1	1	1	1
[3]			9	9	9
[4]					
[5]					
[6]				6	6

4. 체이닝(chaining)

⦿ 오버플로우 문제를 연결 리스트로 해결

☞ 각 버켓에 고정된 슬롯이 할당되어 있지 않음

☞ 각 버켓에, 삽입과 삭제가 용이한 연결 리스트 할당

☞ 버켓 내에서는 연결 리스트 순차 탐색

⦿ (예) 크기가 7인 해시테이블에서

☞ h(k)=k mod 7의 해시 함수 사용

☞ 입력 (8, 1, 9, 6, 13) 적용

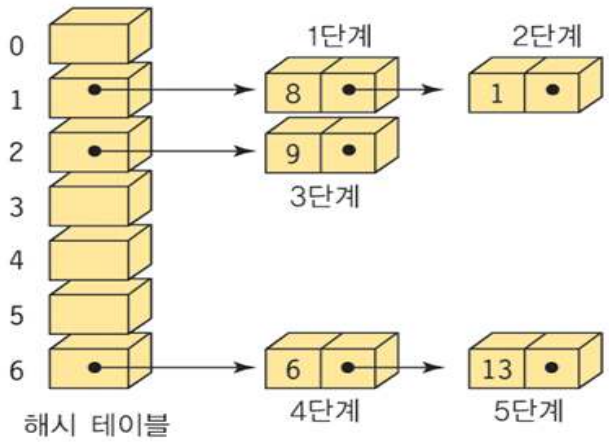
1단계 (8) : h(8) = 8 mod 7 = 1(저장)

2단계 (1) : h(1) = 1 mod 7 = 1(충돌발생->새로운 노드 생성 저장)

3단계 (9) : h(9) = 9 mod 7 = 2(저장)

4단계 (6) : h(6) = 6 mod 7 = 6(저장)

5단계 (13) : h(13) = 13 mod 7 = 6(충돌 발생->새로운 노드 생성 저장)



학습내용2 : 스트링 매칭

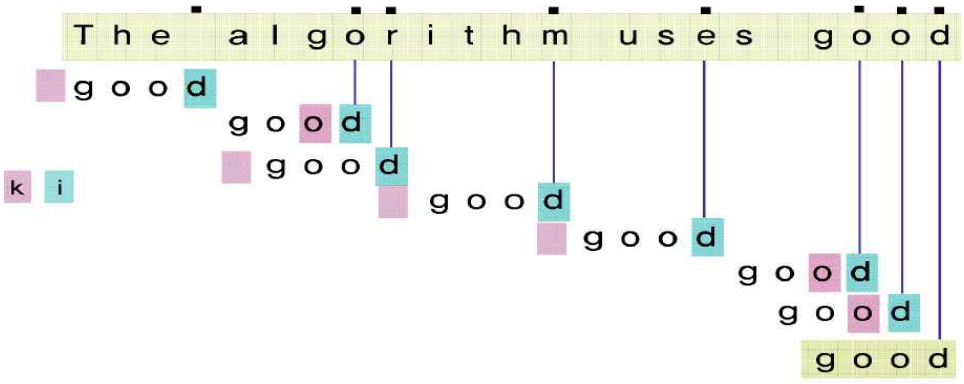
주어진 텍스트에서 주어진 패턴이 어디에 나타나지를 알아내는 문제이다. 예를 들어서 어떤 문서에서 특정 단어를 찾을 때 이것이 스트링 매칭이 되는데, 여기서 문서를 텍스트라 하며 찾아낼 특정단어를 패턴(Pattern)이라 한다.
텍스트 T[1..n]에서 패턴 P[1..m]과 일치하는 모든 부분을 찾아내는 문제이다.
스트링 매칭에는 직선적, 라빈-카프, KMP, 보이어무어(BM)알고리즘 등이 있다.

방법	실행시간	특징
직선적 알고리즘	O((n-m+1)m)→O(mn)	텍스트와 패턴이 비슷할수록 비교 회수가 증가(T의 매 위치마다 P의 모든 문자를 비교 → 최악)
라빈-카프	O((n-m+1)m), O(n+m)일 가능성이 높음	스트링을 해시값으로 계산하여 매칭 (호너의 방법)
KMP	O(n+m)	패턴 전처리 → 최대접두부 이용
보이어-무어(BM)	최악: O((n-m+1)m+|Σ|) 최선: O(m+n/m+|Σ|)	직선적 알고리즘과 유사(단, 문자 비교 방향 : 우측에서 좌측) 일치접미부 방책, 불일치 문자 방책

1. 보이어-무어 알고리즘

보이어-무어 알고리즘은 Boyer와 Moore가 고안한 알고리즘으로서 앞으로 간단히 BM 알고리즘이라고 하겠다. BM 알고리즘은 패턴이 텍스트에 정렬된 각 위치에서 문자의 비교를 좌측에서 우측이 아니라 우측에서 좌측으로 행해나가는 방법이다.
BM 알고리즘이 단순히 우에서 좌로 문자 비교를 행한다면 직선적 알고리즘보다 더 낫다고 볼 수 없으나 BM 알고리즘은 다음과 같은 '일치 접미부(good suffix) 방책'과 '불일치 문자(bad character) 방책'을 도입함에 의해 성능을 개선시켰다.

불일치 문자(bad character) 방책 예



학습내용3 : 기하 알고리즘

1. 기하알고리즘(geometric algorithm)

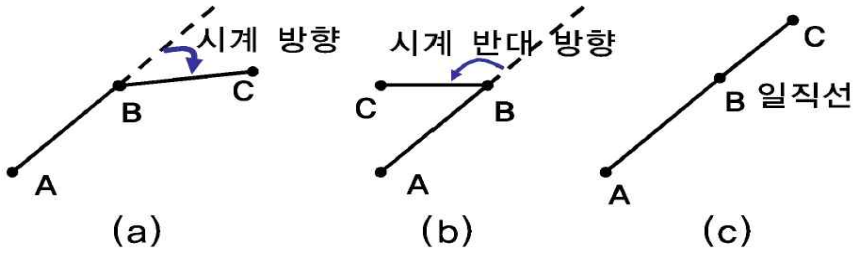
기하 문제를 푸는 알고리즘으로서 점, 선, 다각형과 관련되어 있는 그와 같은 문제를 풀기 위한 알고리즘으로 아래와 같은 내용을 살펴본다.

☞ 두 선분이 서로 교차하는지?
☞ 여러 개의 점들을 꼭지점으로 하는 단순 폐쇄 다각형 만들기.
☞ 주어진 점이 다각형 내부에 존재 하는지?
☞ 주어진 점들을 둘러싸는 가장 작은 볼록 다각형.
☞ 주어진 점들 중에서 최단 경로 찾기.

* 두 선분의 교차 검사

두 선분이 주어졌을 때, 이 둘이 서로 교차하는지 아닌지를 검사하는 것은 가장 기본적인 기하 알고리즘 중의 하나이다. 여기서 두 선분이 교차한다는 것은 그 두 선분이 적어도 한 점을 서로 공유함을 말하는 것 이다.

* 꺾은선 ABC의 꺽이는 방향

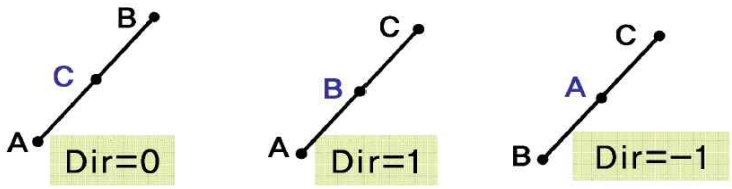


두 점 C,D가 선분 AB를 기준으로 하였을 때 서로 반대편에 있으려면?
➔ 꺾은선 ABC 방향 ≠ 꺾은선 ABD의 방향

* 꺾은선 ABC의 방향을 결정하는 함수를 정의 (Direction[A, B, C])

⦿ A, B, C가 일직선상에 있지 않을 때, 꺾은선 ABC의 방향
  ☞ 시계 방향인 경우 : 1
  ☞ 시계 반대 방향인 경우 : -1

⊙ 점 A, B, C가 일직선상에 있을 때
- ☞ C가 가운데 있는 경우 : 0
  C=A 또는 C=B인 경우
- ☞ B가 가운데 있는 경우 : 1
- ☞ A가 가운데 있는 경우 : -1

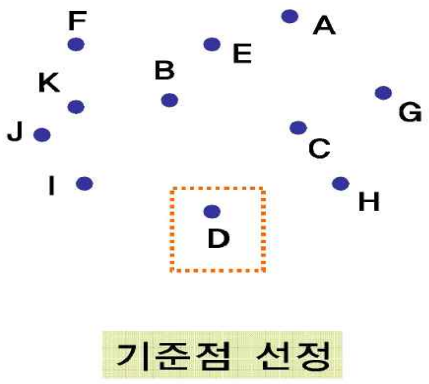


2. 단순 폐쇄 경로

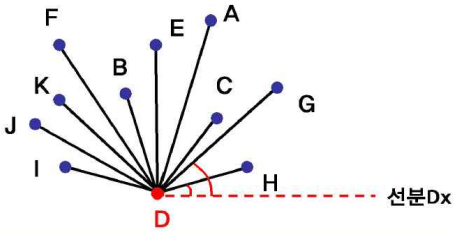
임의의 기준점으로부터 다른 각각의 점까지의 각도를 구한 후 올림차순으로 정렬한 다음 기준점부터 정렬된 순서대로 이으면 그것이 단순 폐쇄경로가 된다. 기준점이 다르면 생성된 단순 폐쇄경로도 다르다.
예를 들어 한명의 택배원이 택배물을 배달하는데 하루에 동안 n개의 집을 방문 할 때 가장 효율적인 경로를 생각 해 보면 택배원이 방문한 집들을 순서대로 직선으로 연결 하여 교차 하지 않도록 하면 한번통과 한 지점은 다시 통과 하지 않으므로 최적의 경로에 가까울 것이다. 이렇게 여러 개의 점이 주어졌을 때 이를 꼭지점으로 하면서 변이 서로 교차하지 않도록 하는 것을 단순 패쇄 경로라 한다.

* 단순 다각형 구하는 과정

과정 1)

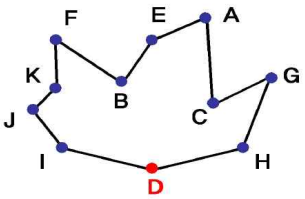


과정 2)



1. 기준점과 그 외의 각 점을 선분으로 연결
2. 선분 Dx와 각 선분과의 각도 계산
3. 각도를 기준으로 점들을 오름차순으로 정렬

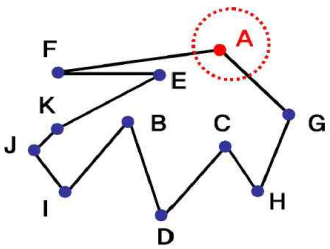
과정 3)



기준점으로부터 시작하여 정렬된 점들을 순서대로 직선으로 연결하고 나서 다시 기준점으로 돌아옴 → 단순 폐쇄 경로, 단순다각형

과정 4)

기준점이 다르면 생성된 단순 폐쇄경로도 다르다.



기준점 D가 아니라 A로 선정한 경우에는 다른 모양의 단순 다각형을 형성

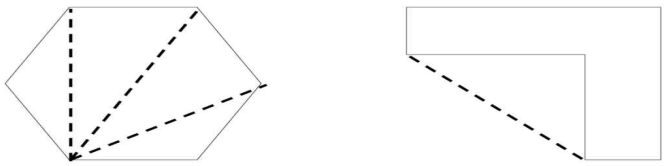
3. 볼록 껍질 찾기

임의의 집합 X에 대한 볼록 껍질이란 X를 포함하는 가장 작은 볼록 집합을 말하며 여러 개의 점이 주어졌을 때 그 점들을 모두 포함하는 가장 작은 볼록 다각형을 그 점집합의 볼록 껍질 (Convex Hull)이라고 한다. 볼록 껍질에는 짐꾸러기, 그레이엄 알고리즘이 있으며, 평균 $O(n^2)$의 실행시간을 갖는다.

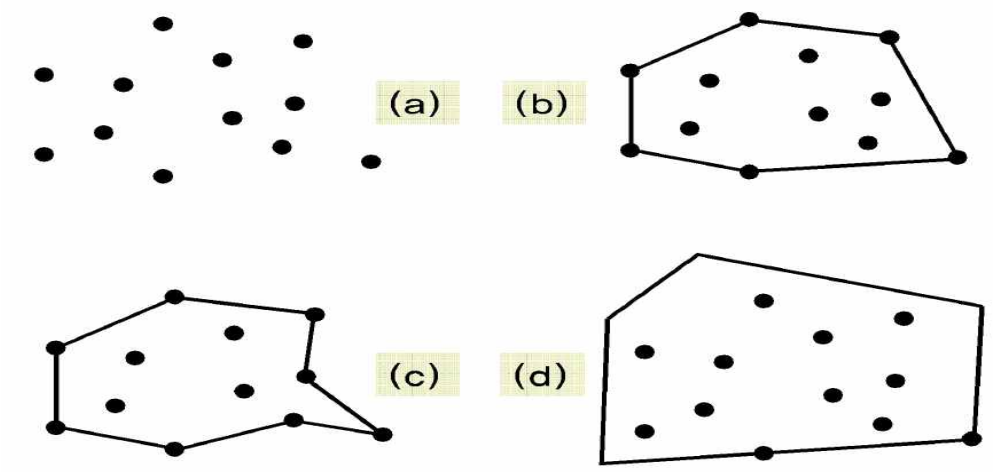
⦿ 볼록 다각형
☞ 다각형 내부의 임의의 두 점을 연결하는 선분이 반드시 다가형 내부에 존재하는 다각형.

⦿ 볼록 껍질(Convex Hull)
☞ 점집합의 모든 점을 포함하는 최면적의 볼록 다각형
☞ 주어진 점들을 모두 둘러싸는 최소길이의 경로.

⦿ 볼록 껍질의 꼭지점
☞ 볼록 껍질 외부의 임의의 직선을 껍질 쪽으로 접근시킬 때 이 직선이 처음으로 만나는 점.



⦿ 점과 볼록 껍질



(a)에 주어진 점집합에 대하여서 (b)는 볼록 껍질이다. 하지만 (c)는 오목 다각형이어서 볼록 껍질이 될 수 없으며 (d)는 최소의 볼록 다각형이 아니므로 볼록 껍질이 될 수 없다. 볼록 껍질은 점들을 모두 둘러싸는 최소 면적의 볼록 다각형이며 볼록 껍질의 꼭지점은 반드시 주어진 점들의 일부이어야 한다. 꼭지점이 아닌 점들은 볼록 껍질의 내부에 속하여야 한다.

4. 짐꾸리기 알고리즘이란?

일명“Jarvis'march”또는“Package wrapping”또는“Gift wrapping”이라 하며 무한대에서부터 임의의 각도로 직선을 점집합 쪽으로 접근 시켜서 직선과 처음 만나는 점들로 볼록 껍질을 형성하는 방법을 의미한다.

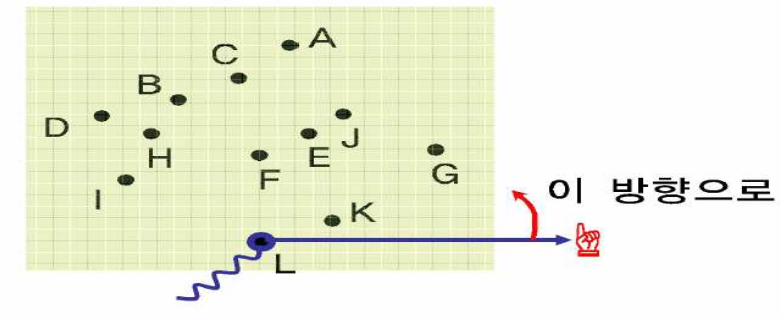
⦿ 볼록 껍질의 특징 중의 하나는 껍질 외부의 임의의 직선을 껍질 쪽으로 접근시킬 때 이 직선이 만나는 것은 볼록 껍질의 한 꼭지점이나 한 변이된다.

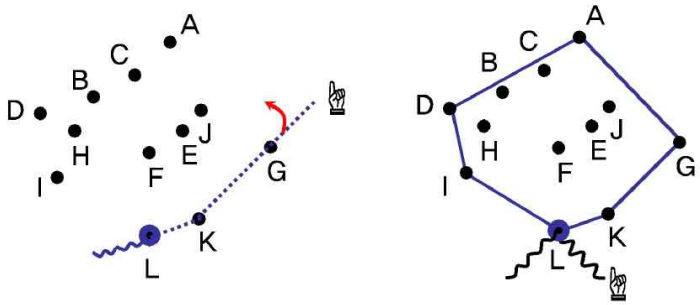
⦿ 무한대에서부터 임의의 각도로 직선을 점집합 쪽으로 접근시켜서 직선과 처음으로 만나는 점들로 구성된 다각형을 만들어도 볼록 껍질이 되는데 이러한 사실에 기초하여 만들어진 것이 짐꾸리기 알고리즘이다.

* 짐꾸리기 알고리즘 적용 예
짐꾸러기 알고리즘의 원리를 못과 테이프를 이용하여 테이프로 둘러싸는 문제를 생각해 보자 여기에 박혀있는 모든 못들이 외부에서 보았을 때 하나도 보이지 않도록 이 테이프로 제일 아래쪽의 못을 일단 테이프로 묶고, 다음 테이프를 수평하게 당긴 다음 시계반대 방향으로 다음 못을 만날 때까지 돌린다. 다음에는 또 다시 이 못에서 다음 못을 만날 때까지 시계반대 방향으로 돌리고 이러한 과정을 처음의 못에 다시 돌아올 때까지 계속 반복하면 이 테이프가 모든 못들을 둘러싸게 된다.
못을 점으로, 테이프를 선으로 바꾸어 생각하면, 방금 설명한 방식이 볼록 껍질을 구하는 방법이 된다. 이 방법에서 시계반대 방향으로 돌려서 만나는 최초의 못은 현재의 못으로부터 가장 작은 각도의 못에 해당한다. 짐꾸리기 알고리즘이 바로 각도를 계산하여 최소각을 선택하는 방식으로 볼록 껍질을 구하는 알고리즘이다. 짐꾸리기 알고리즘을 나타내 보면 다음과 같다.

양의 X축 방향과 이루는 각도를 계산하여 최소의 각을 갖는 점을 다음의 꼭지점 및 기준점으로 취한다.

단계 1: Y 좌표가 최소인 점을 최초의 꼭지점(기준점)으로 선택한다.
단계 2: 기준점으로부터 아직 선택이 안 된 모든 점들에 대한 각도를 계산한다.
단계 3: 최소각을 갖는 점을 다음의 볼록 껍질의 꼭지점(기준점)으로 선택한다.
단계 4: 지금 선택된 꼭지점이 최초의 기준점이면 계산 끝, 아니면 그 점을 기준점 으로 하여 단계 2부터 반복한다.





현재 점과 최소각의 점을 다음 꼭지점으로 선택

* 점 배열 내의 자료의 이동 상황.

단계 \ 인덱스	0	1	2	3	4	5	6	7	8	9	10	11	12
0		B	C	D	E	F	G	H	I	J	K	L	
1	L	B	C	D	E	F	G	H	I	J	K	A	L
2	L	K	C	D	E	F	G	H	I	J	B	A	L
3	L	K	G	D	E	F	C	H	I	J	B	A	L
4	L	K	G	A	E	F	C	H	I	J	B	D	L
5	L	K	G	A	D	F	C	H	I	J	B	E	L
6	L	K	G	A	D	I	C	H	F	J	B	E	L

* 짐꾸리기 알고리즘의 분석

⦿ 실행 시간
☞ 점의 각도를 계산하는 횟수로 계산,

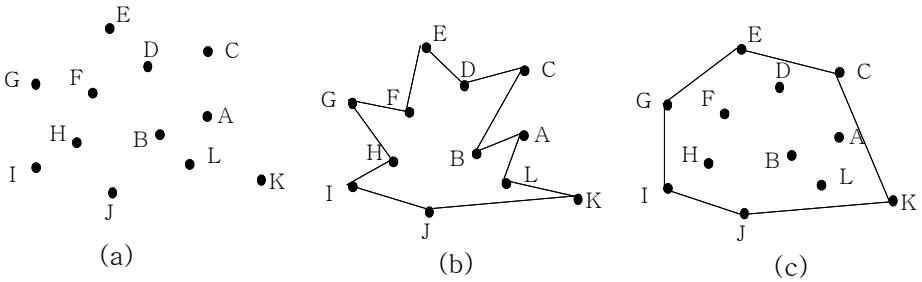
⦿ 최악의 경우
☞ 모든 점들이 볼록 껍질의 점으로 되는 경우.
☞ 수행시간 $O(n^2)$

5. Graham 의 볼록 껍질 구하는 알고리즘

그레이엄 알고리즘은 주어진 점집합으로부터 우선 단순 폐쇄 경로를 구한 다음에 볼록 껍질이 될 수 없는 점들을 제거해 나가는 방법으로 볼록 껍질을 찾는 방법이다.

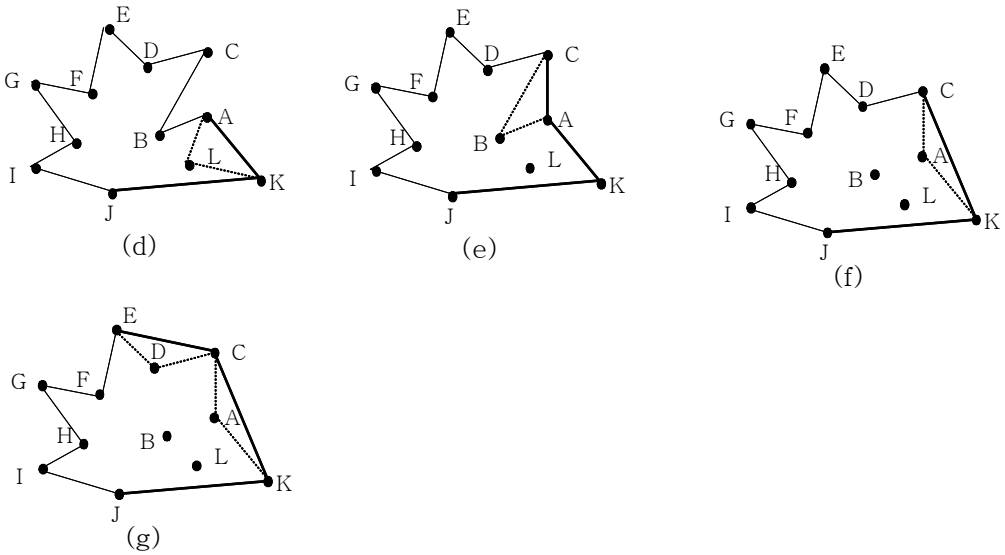
⦿ 점 제거 기준
☞ 어떤 기준점으로부터 단순 폐쇄 경로를 따라 나가면 항상 시계반대 방향으로 꺾이게 되는데, 도중에 시계방향으로 꺾이는 선분이 있다면 선분의 한 점은 볼록 다각형 내부에 존재하는 점이기 때문에 제거 한다.

⊙ 그레이엄 알고리즘의 적용 예



[그림] 볼록 껍질을 만들어 가는 과정

즉 (b)의 꼭지점을 따라가면서 (c)의 볼록 껍질을 만들어야 하는 것이다. 이는 단순폐쇄 경로를 따라 볼록 껍질의 꼭지점이 될 수 없는 점들을 제거해야 함을 말한다. 따라서 이 예의 경우 L, A, B, …등이 제거되어야 하며 제거의 기준은 경로를 따라 점들을 방문해 나갈 때 회전 방향이 시계 방향으로 바뀌면 바로 앞에 방문한 점을 제거하는 것이다. 예를 들어 J가 기준점인 경우, K, L, A, B,… 등의 순서로 점들을 방문해 나갈 것이다. 이제 J, K, L로 해서 A를 방문한다면 회전 방향이 시계 방향으로 바뀌게 된다. 이러한 경우 바로 앞의 L을 볼록 껍질의 후보에서 제거하면 된다. 이렇게 해서 A에 도달한 시점에서는 볼록 껍질의 후보는 다음 그림과 같이 J, K, A이다. A와 K를 잇는 선 안쪽에 L이 존재함을 알 수 있게 된다.



* 볼록 껍질을 만들어 가는 과정

이제 다음 B는 시계반대 방향이므로 그대로 두고, 다음 C를 방문한다. 그러면 회전방향이 시계방향으로 바뀌게 되고 따라서 B를 제거하게 됨으로 그림 (e)의 상태처럼 만들어지게 된다. 이 상태에서 다시 KAC의 회전 방향이 시계 방향이므로 A 또한 제거 되어야 하며 이것이 그림 (f)상태가 된다. 이러한 과정이 J에게 돌아올 때까지 반복하면 볼록 껍질이 만들어지게 된다.

* 그레이엄 알고리즘의 분석
⦿ 단순 폐쇄 경로 → O(nlogn)
☞ 기준점 선정 O(n)
☞ 각도 계산 O(n)
☞ 각도에 따른 오름차순 정렬 O(nlogn)

⦿ 점 제거 → O(n)

【학습정리】

1. 해싱
키 값에 대한 산술적 연산에 의해 탐색키를 입력받아 해시 주소(hash address) 생성 테이블의 주소를 계산하여 항목에 접근하는 방법이다.
* 해시함수구하는 방법
☞ 제산 함수
☞ 폴딩 함수
☞ 중간제곱 함수
☞ 비트추출 함수
☞ 숫자 분석 방법
* 충돌해결방법
☞ 선형조사법: 충돌이 일어난 항목을 해시 테이블의 다른 위치에 저장
☞ 체이닝: 각 버켓에 삽입과 삭제가 용이한 연결 리스트 할당

2. 볼록 껍질 찾기
- 볼록 껍질이란 점집합의 모든 점을 포함하는 최소 면적의 볼록다각형
- 볼록 껍질을 찾는 방법: 단순한 방법, 짐꾸리기 알고리즘, 그레이엄 알고리즘

3. 짐꾸리기 알고리즘
- 무한대에서부터 임의의 각도로 직선을 점집합쪽으로 접근시켜서 직선과 처음 만나는 점들로 볼록 껍질을 형성하는 방법, $O(n^2)$
- 방법
① Y좌표가 최소인 점을 최초의 꼭지점(기준점)으로 선택
② 기준점으로부터 아직 선택이 안 된 모든 점들에 대한 각도를 계산한다.
③ 최소각을 갖는 점을 다음의 볼록 껍질의 꼭지점(기준점)으로 선택한다.
④ 지금 선택한 꼭지점이 최초의 기준점이면 계산을 종료하고, 아니면
그 점을 기준으로 단계②부터 반복

4. 그레이엄 알고리즘
- 주어진 점집합으로부터 우선 단순 폐쇄 경로를 구한 후 볼록 껍질의 꼭지점이 될 수 없는 것을 제거해 나가는 방법.
- 점 제거 방법
→ 볼록 다각형의 꼭지점을 어떤 기준점으로부터 반시계 방향으로 따라가면 항상 반시계 방향으로 꺾인다.
→ 이와 같이 단순 폐쇄 경로를 따라가는 도중에 꺾은선 ABC의 방향이 시계 방향이면 점 B를 제거한다. 왜냐하면 점B는 그때까지 만들어진 볼록 다각형의 내부에 존재하는 점이기 때문이다.