자료구조 기말고사

서울대학교, 언어학과

2021-12659, 박유나

1. 1차 군집화(Primary clustering)와 2차 군집화(Secondary clustering) 이슈를 해결한 해시 (Hash) 기반 검색 성능은, O(1)라는 극단의 효율을 보입니다. 1차 군집화와 2차 군집화가 발생하는 이유를, ① 선형 조사(Linear probing)와 ② 이차원 조사(Quadratic)라는 용어를 사용하여 각각 기술하세요. (5점)

해시 기반 데이터 구조는 해시 함수(h(x))를 이용해 데이터 x로부터 인덱스를 구하여 저장하는 방식을 사용합니다. 다른 데이터들이 해시 함수의 결과에 따라 동일한 인덱스를 갖게 되는 현상을 해시 충돌이라고 하며, 충돌이 발생할 때 해시 테이블 내의 다른 위치를 찾아 데이터를 저장하는 방식을 Colsed hasging이라 합니다. Closed hashing은 충돌한 인덱스를 갖는 데이터를 옮길 인덱스를 찾는 방법에 따라 Linear hashing과 Quadratic hashing으로 분류할 수 있습니다. 각 방식에 따라 군집화가 발생할 수 있습니다. 군집화란 데이터가 특정 위치에 몰려 있는 현상을 의미하며, 각각의 원인은 해시 탐사 방법인 Linear hashing과 Quadriatic hashing 특성에 기인합니다.

먼저 Linear hashing에서는 충돌이 발생할 때, 해시 테이블의 다음 위치를 순차적으로 탐색합니다. 즉 h(x)인 index에서 충돌이 발생하면, [h(x)+i mod array_size] 의 인덱스에 데이터를 저장합니다.(i는 i번째 충돌을 의미) 이러한 방식은 연속적인 인덱스를 생성하기 때문에 충돌이 발생할 때마다 새로 추가된 요소들이 근처에 저장되므로 데이터가 군집처럼 모여있게 됩니다. 같은 충돌이 발생할 때마다 기존의 군집을 연속적으로 탐색하며 빈 공간을 찾게 되므로 군집은 점점 커지고 충돌은 빈번해지는 현상이 발생합니다. 이렇게 군집이 생성되는 현상을 1차 군집화라합니다.

다음으로 Quadratic hashing에서는 충돌이 발생할 때, i에 관한 이차함수의 보폭으로 점프한 인덱스에 저장하려 합니다. 즉 h(x)인 index에서 충돌이 발생하면, [h(x)+ ci^2 + c2i mod array_size] 의 인덱스에 데이터를 저장합니다. 이 방법을 사용하면 특정 영역에 데이터가 몰리는 1차 군집이 발생하더라도 큰 보폭으로 움직이므로 1차 군집을 빠르게 벗어날 수 있습니다. 그러나 여러 원소가 동일한 초기 해시값을 갖게 되면, 똑같은 점프 시퀀스를 따라 저장되므로 제곱의 패턴에 따라 데이터의 군집이 형성될 수 있으며, 이를 2차 군집화라 합니다.

2. Kruskal 알고리즘과 Prim 알고리즘은, 주어진 그래프로부터 최소신장트리 (Minimum spanning tree) 도출을 수행합니다. Kruskal 알고리즘 대비 Prim 알고리

증을 활용하여 최소 신장트리를 도출하는 접근법이 보다 효율적인 경우와 그 이유를 서술하세요. (5점)

Kruskal 알고리즘보다 Prim 알고리즘이 효율적인 경우를 설명하기 위해서는 각 알고리즘의 작동원리와 시간복잡도에 대해 이해할 필요가 있습니다.

Kruskal 알고리즘을 모든 edge를 가중치에 따라 오름차순으로 정렬 후, 가장 작은 weight를 갖는 edge부터 시작하여, 사이클을 형성하지 않도록 edge를 선택하는 알고리즘입니다. 사이클이 형성되었는지 확인하기 위해서는 Union find 알고리즘을 사용합니다. Kruskal 알고리즘의 시간복잡도를 계산하기 위해서는 edge 정렬과 사이클을 확인하는 과정을 고려해야합니다. 먼저 edge를 가중치에 따라 오름차순으로 정렬하는 과정은 정렬 알고리즘의 시간복잡도에 따르는데 일반적으로 퀵 정렬이나 병합정렬 같은 알고리즘을 사용할 경우에 O(ElogE)입니다. 사이클을 확인은 edge마다 이루어지고 union find의 시간복잡도는 O(a(v))라고 하므로, O(Ea(v))의 시간복잡도를 갖습니다. 그런데 a(v)는 거의 상수에 가깝다고 하는 조사 결과에 따라 최종적인 사이클 확인의 시간복잡도는 O(E)라 할 수 있습니다. 최종적인 Kruskal 알고리즘의 시간 복잡도는 O(ElogE)입니다.

반면 Prim알고리즘은 임의의 vertex에서 시작하여, 현재까지 연결된 vertex들에서 연결되지 않은 vertex들에 대해, 가장 가중치가 작은 vertex를 연결하는 vertex 선택 기반으로 동작합니다. Prim 알고리즘의 시간 복잡도는 모든 vertex에 대해 탐색을 진행하므로 O(V)이고 각 vertex마다 최소 weight를 갖는 edge를 탐색해야하므로 추가로 O(V)가 소요됩니다. 따라서 시간 복잡도는 O(V^2)입니다. 만약 자료구조가 priority queue라면 각 vertex에서의 최소 edge를 찾는 소요되는 비용이 줄어드므로 O(VlogV)의 시간이 소요되고 추가로 각 vertex의 인접한 간선을 찾는 시간이 E가 필요하므로 O((V+E)logV)이 됩니다. 일반적으로 E가 V보다 크기 때문에 최종적인 Prim 알고리즘의 시간복잡도는 O(ElogV)라 할 수 있습니다.

결론적으로 시간복잡도를 비교하였을 때 Kruskal은 O(ElogE), Prim은 O(ElogV)이므로 E의 크기가 큰 그래프, 다시 말해 edge가 많은 그래프의 경우에 Prim 알고리즘이 효율적입니다.

3. 활용되는 데이터 구조에 따라, 동일한 알고리즘이라도 시간 복잡도가 상이할 수 있는 사례 한 가지를 제시하세요. (5점)

위에서 언급한 Prim알고리즘을 들 수 있습니다. Prim 알고리즘에서 자료구조가 배열이라면 시간복잡도가 O(V^2)가 됩니다. 그 이유는 모든 vertex에 대해 탐색을 진행하므로 O(V)이고 각 vertex마다 최소 weight를 갖는 edge를 탐색해야하므로 추가로 O(V)가 소요되기 때문입니다. 하지만 자료구조로 priority queue를 활용하면 시간복잡도가

O(ElogV)로 낮아집니다. priority queue는 각 원소에 대해 우선순위를 할당하고, 우선순위가 가장 높은 원소를 빠르게 추출할 수 있는 데이터 구조입니다. Prim 알고리즘에서 priority queue가 사용되면 최소 edge을 빠르게 선택할 수 있습니다. 이를 통해 현재 MST에 포함된 vertex과 연결된 edge 중 가장 작은 가중치의 edge를 선택하는 과정을 O(logV)의 시간내로 빠르게 찾을 수 있습니다. 따라서 Priority Queue를 사용하면 V개의 vertex에 대해 각 vetex 마다 O(logV) 시간이 소요되며, E개의 edge에 대해서도 인접한 edge에 따라 각 edge의 가중치를 O(logV) 시간에 업데이트할 수 있기 때문에 Prim 알고리즘의 시간복잡도는 O((V+E)logV)이 됩니다. 그리고 위에서 언급했듯 일반적으로 E가 V보다 크기 때문에 최종적인 Prim 알고리즘의 시간복잡도는 O(ElogV)라 할 수 있습니다. 이처럼 자료구조에 따라 알고리즘의 시간복잡도가 달라질 수 있습니다.

4. 특정 조건을 만족하는 문제에 동적 프로그래밍(Dynamic programming)을 적용할 경우, 시간 복잡도 측면에서 높은 효율 증대를 기대할 수 있습니다. 동적 프로그래밍을 적용할 수 있는 문제의 조건들을 기술하세요.

동적 프로그래밍은 복잡한 문제를 더 간단한 부분 문제로 나누어 해결하는 방법입니다. 반복적으로 발생하는 부분 문제를 해결할 때 시간 복잡도를 크게 줄임으로써 전체 문제를 효율적으로 해결할 수 있습니다. 동적 프로그래밍을 적용할 수 있는 문제는 Overlapping Subproblems와 Optimal substructure라는 두 조건을 만족해야 합니다:

먼저, Overlapping Subproblems는 문제를 해결하는 과정에서 동일한 부분 문제가 여러 번 계산되는 경우를 말합니다. 즉, 문제를 해결하기 위해 동일한 부분 문제를 반복적으로 계산해야 하는 경우입니다. 예를 들어 피보나치 수열에서는 F(n) = F(n-1) + F(n-2) 관계를 사용하여 하위 문제를 여러 번 계산합니다. 만약 F(3) = F(2) + F(1)로 F(3)을 계산했다면, 이후에 F(4) = F(3) + F(2)를 계산하기 위해 F(3)을 다시 사용할 수있습니다.

다음으로 Optimal substructure는 전체 문제의 최적 해결책이 부분 문제들의 최적 해결책으로 구성될 수 있는 경우를 말합니다. 즉, 문제를 해결하기 위해 부분 문제들을 해결하면, 전체 문제의 최적 해결책을 얻을 수 있습니다. 예를 들어 피보나치 수열에서 n번째 항을 계산하기 위해서는 F(n−1)과 F(n−2)를 계산해야 합니다. 달리 말하면 F(n−1)과 F(n−2) 합을 해결하면 전체 문제인 F(n)의 최적값을 얻을 수 있으므로 F(n−1)과 F(n−2)의 최적값을 결정하는 것이 부분 문제라 할 수 있습니다.

5. 최대 이윤 행렬 경로(Maximum profit path in matrix) 문제는, 해결책 행렬(Solution matrix)을 활용하는 tabulation 형태의 동적 프로그래밍을 적용할 경우 효율적으로 해결될 수 있습니다. 문제 행렬을 P 그리고 해결책 행렬을 S라고 할

때, 해결책 행렬의 (i, j) 요소는 S(i, j) = P(i, j) + max[S(i-1, j), S(i, j-1)]로 표기되는데요, S(i, j)의 의미를 서술하세요.(5점)

* 최대 이윤 행렬 경로 문제: m×n 형태로 구성된 2차원 행렬을 대상으로, 해당 행렬의 (1, 1)에서 시작하여 (m, n)에 이르는 모든 경로 중, 이윤을 최대화하는 경로를 도출하는 문제로, 이동 가능 방향은 좌에서 우 그리고 위에서 아래로 한정.

해결책 행렬 S(i,j) 의 의미는, (1,1) 에서 (i,j) 까지 도달할 때의 최대 이윤을 나타냅니다. 그 이유는 S(I, j)를 계산하는 방식에 있습니다. P(i,j)는 행렬의 (i,j) 위치에서 얻을 수 있는 이윤을 나타내며, S(i−1,j)는 현재 위치의 위쪽 위치에서 최대 이윤을, S(i,j−1)는 현재 위치의 왼쪽 위치에서 최대 이윤을 의미합니다. 최대 이윤 행렬 경로 문제에 tabulation방식의 동적 프로그래밍을 적용하면 해결책 행렬을 순차적으로 채워 나가면서 문제를 해결합니다. 이를 통해 각 하위 문제 S(i−1,j)와 S(i,j−1) 해결하고, 이 하위 문제들의 해결 결과를 재사용하여 전체 문제를 해결할 수 있습니다.

S(i, j) = P(i, j) + max[S(i-1, j), S(i, j-1)]로 표기되는데, max[S(i−1,j),S(i,j−1)]는 현재 위치 (i,j)에 도달하기 위해 이전에 도달할 수 있는 두 위치 중에서 더 높은 이윤을 가진 위치를 선택함을 말합니다. 이 방법을 통해 전체 행렬에서 최대 이윤 경로를 효율적으로 찾을 수 있습니다. 즉, S(i, j)는 (i, j) 위치에서 얻을 수 있는 이윤 P(i,j)와 그 위치에 도달하기 위해 가능한 최적의 경로 중 가장 높은 이윤을 가진 경로의 이윤을 합산한 값이므로 (1,1) 에서 (i,j) 까지 도달할 때의 최대 이윤을 나타낸다고 할 수 있습니다.

6. 문제 복잡도를 나타내는 부류(Class)인 NP, P, NP-hard 그리고 NP-complete을 각각 설명하고, 각 부류 간의 관계를 기술하세요. (5점)

P는 P는 다항 시간 내에 해결할 수 있고 주어진 해를 다항시간 내에 검증가능한 문제의 집합입니다. 다항시간 내에 풀 수 있다는 의미는, 시간 복잡도가 O(n^2), O(n^3)같이 임의의 상수인 k에 대해 O(n^k)로 표현될 수 있다 는 의미입니다. 대표적인 예로는 버블 정렬이나 퀵정렬 같은 정렬알고리즘이 있습니다.

어떠한 문제를 해결하는 Non-deterministic Polynomial Time을 가진 알고리즘이 존재한다면 해당 문제의 집합을 NP 라고 한다. NP Problem은 다항식 시간내에 해결할 수 있는지 없는지 모르는 문제들입니다. 즉 만약 해가 주어진다면 주어진 해를 다항시간 내에 검증가능한 문제의 집합입니다. 대표적인 예로는 SAT(여러 개의 절(clause)로 구성된 논리식에서 모든 절이 참이 되도록 만드는 변수 할당을 찾는 문제)가 있습니다.

NP-hard는 NP에 속한 모든 문제를 다항 시간 내에 'A' 문제로 Reduction 할 수 있는 경우 'A' 문제를 NP-hard라고 합니다. Reduction이란 어떤 문제를 다른 문제로 변환하는

과정을 말합니다. 대표적인 예로는 여행하는 세일즈맨 문제(TSP)의 optimization version 이 있고, SAT는 TSP문제로 환원 가능합니다.

NP-complete는 NP에 속하며 동시에 NP-hard인 문제들의 집합입니다. 대표적인 예로는 그래프의 클리크 문제가 있습니다.

다음은 부류간의 관계에 대해 설명하도록 하겠습니다.

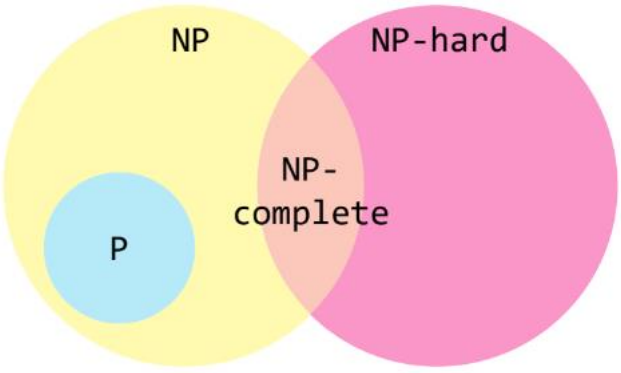
모든 P 문제는 NP 문제입니다. 즉, P⊆NP입니다. 이는 다항 시간 내에 문제를 해결할 수 있다면, 그 해답도 다항 시간 내에 검증할 수 있음을 의미합니다. P=NP 인지 아닌지는 아직 해결되지 않은 난제입니다. 만약 이가 증명된다면, 모든 NP 문제는 다항 시간 내에 해결할 수 있게 됩니다.

NP-complete 문제는 NP에 속하며, NP의 모든 문제를 다항 시간 내에 변환할 수 있는 문제들입니다. 즉 NP- complete⊆NP입니다. 모든 NP 문제는 NP-complete 문제로 환원 될 수 있습니다.

NP-hard 문제는 NP의 모든 문제를 다항 시간 내에 변환할 수 있는 문제들입니다. 하지만 NP-hard 문제는 반드시 NP에 속하지는 않습니다.

NP-complete 문제는 NP에 속하며 NP-hard인 문제들입니다. 모든 NP-complete 문제는 NP-hard 문제입니다. 즉 NP- complete⊆NP- hard입니다.

4가지의 부류를 도식화 하면 아래와 같습니다.



1. 김응희, 자료구조, 14주차 수업자료

7. 하단의 SAT(Satisfiability) 문제를 클리크(Clique) 문제로 환원(Reduction)하여 해결책(Solution)을 도출하고, 도출 과정을 함께 서술하세요. (10점)

(x1∨x2)∧(¬x1∨¬x2)∧(x1∨x3)

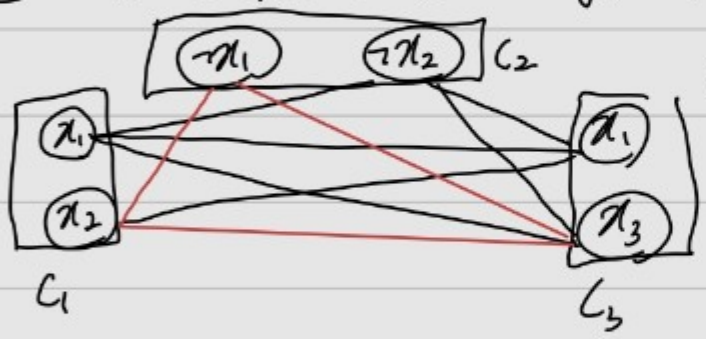
7. ① SAT 문제에 절과 노드를 분석한다.

$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_3)$

$= C_1$ $= C_2$ $= C_3$

또욱 $\Rightarrow x_1, x_2$ $\Rightarrow \neg x_1, \neg x_2$ $\Rightarrow x_1, x_3$

② 서로다른 절에 속한 노드 간에 edge 추가한다. (이때, 동일한 변의 공정, 부정 (ex $x_1, \neg x_1$



③ 각절에서 하나씩 노드를 선택해 모

잡는다.

($\neg x_1$=True)

$\neg x_1$, x_2, x_3 는 3 Clique를 형성한다. 따라서 x_1 = fake, x_2= True, x_3=

④ Clique 방식으로 발견한 solution이 SAT 문제를 만족하는지 검증하여 본

검증. (F∨T) ∧ (T∨F) ∧ (F∨T)

= T ∧ T ∧ T ⇒ True. 이므로