DGR-Tree를 위한 KNN 검색 알고리즘

이득우*, 강홍구*, 한기준*
*건국대학교 컴퓨터·정보통신공학과
e-mail:{dwlee, hkkang, kjhan}@db.konkuk.ac.kr

A K-Nearest Neighbor Search Algorithm for DGR-Tree

Deuk-Woo Lee*, Hong-Koo Kang*, Ki-Joon Han*
*Dept of Computer & Information Communication Engineering, Konkuk University

요 약

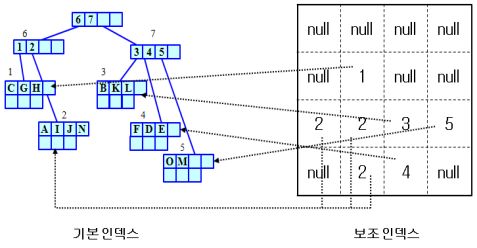
유비쿼터스 컴퓨팅 환경에서의 LBS에서는 점차 대용량화 및 밀집화 경향을 보이는 POI에 대한 빠른 KNN 검색이 중요하다. 따라서 본 논문에서는 기존의 DGR-Tree를 위해서 POI에 대한 빠른 KNN 검색을 위한 KNN 검색 알고리즘을 제시하고, 또한 성능 평가를 통해 그 우수성을 입증한다.

1. 서론

유비쿼터스 컴퓨팅 환경의 LBS(Location Based Service)에서는 POI(Point of Interest)가 점차 대용량화 및 특정 지역 밀집화가 심화되는 경향을 보인다[2]. 따라서 POI의 특성을 고려한 인덱스 구조에 대한 연구가 필요하다. 본 논문에서는 사용자의 위치정보와 관련된 POI에 대한 빠른 KNN(K-Nearest Neighbor) 검색을 위하여 기존의 인덱스 구조인 DGR-Tree(Dynamic-level Grid based on R-Tree)[4] 위한 KNN 검색 알고리즘을 제시하였다. 또한 성능 실험을 통해 우수성을 입증하였다.

2. 관련연구

(그림 1)은 DGR-Tree[4]의 예를 보여준다.



(그림 1) DGR-Tree의 예

(그림 1)에서 화면의 왼쪽 그림은 DGR-Tree의 기본 인덱스(이하 DGR-Tree라 칭함)를 보여주며, 오른쪽 그림은 데이타의 밀집도에 따라 구성되는 보조 인덱스인 동적 레벨 그리드를 보여준다. 동적 레벨 그리드를 구성하고 있는 각 셀은 해당 셀과 연관되는 DGR-Tree의 리프 노드를 가리키고 있으며, 이를 통해 인덱스 접근 성능을 크게 향상시킨다.

LBS 환경에서는 KNN 검색 시 트리 기반의 인덱스 및 거리 기반 KNN 검색 알고리즘이 일반적으로 사용된다[3]. 이는 최소 거리와 우선순위 큐를 사용하는 방식으로서 Quad-Tree, KD-Tree, R-Tree 등의 다양한 인덱스 구조에 적용할 수 있다는 장점이 있다. 그러나 기존의 논문에서 제시한 DGR-Tree에 해당 알고리즘을 적용할 경우 DGR-Tree의 다양한 장점을 활용할 수가 없다.

3. DGR-Tree를 위한 KNN 검색 알고리즘

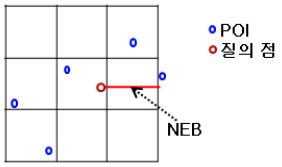
DGR-Tree에서 KNN 검색 알고리즘을 수행하기 위해서는 후보 노드를 관리하는 후보 리스트, 검색이 완료된 노드를 저장하는 검색 완료 리스트, 검색 결과를 저장하는 결과 리스트 등이 필요하다. 후보 리스트는 후보 노드의 Linked-List로 구성되며, 각각의 후보 노드는 질의점과의 최소 거리, 다음에 비교할 엔트리의 위치를 나타내는 옵셋, 후보 노드의 엔트리 개수와 엔트리, 다음 후보 노드의 포인터로 구성된다. 또한 후보 리스트는 후보 노드의 최소 거리를 기준으로 정렬된다. 후보 노드의 최소 거리는 질의점과 후보 노드 MBB 간의 최단 거리로서, 질의점이 해당 객체에 포함되는 경우 최단 거리는 0이 된다.

후보 리스트는 질의점이 위치한 셀과 연결된 DGR-Tree의 리프 노드가 있는 경우 해당 리프 노드를 후보 노드로서 후보 리스트에 추가한다. 현재의 후보 리스트에서 KNN 검색을 완료하지 못한 경우, 질의점이 위치한 셀에 인접된 최대 8개까지의 셀을 검색한다. 이때 검색된 각각의 셀과 연결된 DGR-Tree 리프 노드가 있는 경우 해당 리프 노드를 후보 노드로서 후보 리스트에 추가한다. 만약 8개까지 셀을 확장한 후에도 현재의 후보 리스트에서 KNN 검색을 완료하지 못한 경우, DGR-Tree의 루트 노드를 후보 리스트에 추가한다.

검색 완료 리스트는 검색 완료된 노드 정보를 저장하는 배열 형태로 구성된다. 결과 리스트는 KNN 검색 시 사용자가 요청하는 K개만큼의 결과 엔트리의 배열로 구성된

다. 결과 리스트에 저장되는 결과 엔트리의 자료 구조는 해당 POI와 질의점과의 최소 거리와 OID로 구성된다.

DGR-Tree를 위한 KNN 검색 알고리즘에서는 노드 내에 있는 질의점으로부터의 엔트리의 최소 거리가 다른 노드에 있는 엔트리의 최소 거리보다 큼에도 불구하고 근접 객체로 결정되는 오류를 방지하기 위해서 NEB(Non Extensible Boundary)를 저장한다. NEB는 노드 검색 시 확장을 고려하지 않아도 되는 최소 거리로서 (그림 2)는 NEB의 예를 보여준다.



(그림 2) NEB의 예

NEB는 루트 노드를 검색한 이후에는 불필요하다. (그림 3)은 KNN 검색 알고리즘을 보여준다.

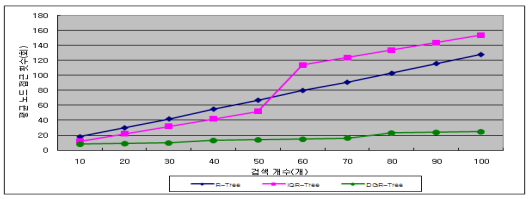
```
Algorithm : KnnSearch(POINT qp, INT k,
DYNAMICGRID dg,  DGRTREE dt )
Begin
1.  RESULT result ← Ø
2.  DONELIST dl ← Ø
3.  CANDIDATELIST clist ← Ø
4.  CELL cell ← FindCell(qp , dg)
5.  IF( cell.node <> NULL )
6.      Insert(cell.node, qp, clist, dl)
7.      Search(qp, clist, dl, result)
8.      IF(  number of result == k )
9.          RETURN result
10.     END IF
11.  END IF
12.  CELL celllist[] ← find set of cell which
adjacent cell of cell
13.  FOREACH  cell  IN celllist
14.      Insert(cell.node, qp, clist, dl)
15.  END FOR
16.  Search(qp, clist, dl, result)
17.  IF(  number of result == k )
18.      RETURN result
19.  END IF
20.  Insert(dt.root, qp, clist, dl)
21.  Search(qp, clist, dl, result)
23.  RETURN result
End
```

(그림 3) KNN 검색 알고리즘

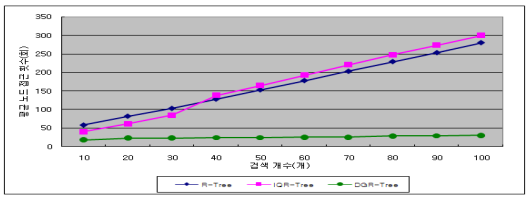
(그림 3)에서 보인 바와 같이 먼저 질의 결과를 저장하기 위해 result, 후보 리스트와 검색 완료 리스트를 저장하기 위한 후보 리스트 객체 clist, 검색 완료 리스트 객체 dl을 선언한다. 다음 FindCell(*qp, dg*) 함수를 호출하여 질의점이 위치한 곳의 셀을 찾아 후보 리스트에 노드를 삽입하기 위해 Insert(*node, qp, clist, dl*) 함수를 호출한다. 그리고 Search(*qp, clist, dl, result*) 함수를 이용하여 현재까지 후보 리스트에서 입력된 노드 데이타를 기반으로 실제 KNN 검색을 수행하게 된다.

4. 성능평가

본 논문에서는 DGR-Tree와 비교 대상으로서 LBS에서 많이 사용되고 있는 R-Tree 및 iQR-Tree[1]를 동일 환경에서 구현하였다. 실험 데이타로서 편향 분포를 가지는 서울시에 존재하는 9만개의 음식점 POI와 40만개의 산업체 POI를 사용하였다. (그림 4)와 (그림 5)는 K를 10에서 100까지 10개씩 증가시키면서 KNN 검색하는 동안 평균 노드 접근 횟수를 보여준다.



(그림 4) 서울시 음식점 POI(편향 분포)



(그림 5) 서울시 산업체 POI(편향 분포)

위 그림에서처럼 DGR-Tree의 KNN 검색 성능이 R-Tree나 iQR-Tree에 비해 월등히 좋은 이유는 동적 레벨 그리드를 통해 트리 검색 없이 데이타가 존재하는 리프 노드에 직접 접근할 수 있기 때문이며, 후보 객체가 있을 가능성이 높은 인접 셀을 먼저 검색하기 때문이다.

5. 결론

본 논문에서 제시한 DGR-Tree를 위한 KNN 검색 알고리즘은 동적 레벨 그리드를 활용하여 질의점에 존재하는 셀 또는 인접 셀을 통해 검색 후보 노드에 직접 접근하여 빠른 응답이 가능하다. 또한 정렬에 따른 오버헤드 및 셀 검색으로 인한 오버헤드를 최소화하고 있다.

참고문헌

[1] Bo, H., and Qiang, W., "A Spatial Indexing Approach for High Performance Location Based Services," The Journal of Navigation, Vol.60, No.1, 2007, pp.83-93.

[2] Frentzos, E., Gratsias, K., and Theodoridis, Y., " Towards the Next Generation of Location-based Services," Proc. of the Intl. Workshop on Web and Wireless Geographical Information Systems, 2007, pp.202-215.

[3] Hjaltason G. R., and Samet H., "Distance Browsing in Spatial Databases," ACM Transactions on Database Systems, Vol.24, No.2, 1999, pp.265-318.

[4] 이득우, 강홍구, 한기준, "DGR-Tree : u-LBS에서 POI의 효율적인 검색을 위한 인덱스 구조", 2009 GIS 공동추계학술대회 논문집, 한국공간정보시스템학회 2009, pp.89-97.