Aula de Ciência da Computação: O Algoritmo Geohash

Introdução

O Geohash é um algoritmo de codificação geoespacial que transforma coordenadas geográficas (latitude e longitude) em uma sequência curta de caracteres alfanuméricos. Essa representação compacta facilita o armazenamento, a indexação e a busca eficiente de dados espaciais, sendo amplamente utilizada em sistemas de informação geográfica (SIG), bancos de dados e aplicações de localização.(GeeksforGeeks)

História e Motivação

O conceito de Geohash foi introduzido por Gustavo Niemeyer em 2008. Ele desenvolveu o sistema como uma maneira de criar URLs curtas que representassem locais geográficos específicos, facilitando o compartilhamento de locais na internet. O site geohash.org foi lançado para demonstrar essa funcionalidade.(Wikipedia)

Embora Niemeyer tenha criado o Geohash de forma independente, ideias semelhantes já haviam sido exploradas anteriormente. Em 1966, G.M. Morton propôs uma técnica de ordenação espacial conhecida como curva Z-order ou Morton order, que intercalava bits de coordenadas para preservar a proximidade espacial em estruturas de dados. No entanto, a proposta de Morton não era voltada para representação textual ou uso em aplicações web. (Graph Search, Wikipedia)

Antes do Geohash

Antes do surgimento do Geohash, a representação e indexação de dados geoespaciais eram realizadas principalmente por meio de coordenadas decimais (latitude e longitude) e estruturas de dados como R-trees. Essas abordagens, embora eficazes, apresentavam desafios em termos de eficiência de busca e complexidade de implementação, especialmente em bancos de dados relacionais que não ofereciam suporte nativo a tipos espaciais.(GIS Stack Exchange)

Funcionamento do Geohash

O Geohash divide o espaço geográfico em uma grade hierárquica de células retangulares. Cada célula é identificada por uma string de caracteres baseada em uma codificação base32 personalizada, que exclui caracteres ambíguos como "a", "i", "l" e "o".(Open Source GIS Data, Wikipedia)

O processo de codificação envolve a interleaving (entrelaçamento) dos bits das coordenadas de latitude e longitude, resultando em uma sequência binária que é então convertida para a representação base32. A precisão da localização é determinada pelo comprimento da string: quanto mais longa, mais precisa. (GeeksforGeeks)

Uma característica importante do Geohash é que locais próximos tendem a ter prefixos semelhantes, o que facilita buscas por proximidade e agrupamento de dados geoespaciais.(GeeksforGeeks)

Vantagens do Geohash

- **Compactação de Dados**: Transforma coordenadas de ponto flutuante em strings curtas, reduzindo o espaço de armazenamento.
- **Hierarquia Natural**: A estrutura hierárquica permite ajustar a precisão conforme necessário, simplesmente truncando ou estendendo a string.
- **Facilidade de Busca**: A similaridade de prefixos entre locais próximos permite buscas eficientes por proximidade usando operações de prefixo em strings.
- **Implementação Simples**: Pode ser facilmente implementado em bancos de dados que não possuem suporte nativo a tipos espaciais, utilizando índices de texto.
- **Domínio Público**: O Geohash é de domínio público, o que facilita sua adoção em diversas aplicações sem preocupações com licenciamento. (GIS Stack Exchange)

Desvantagens do Geohash

- **Distorsão de Área**: Devido à forma esférica da Terra, as células do Geohash não têm tamanho uniforme em todas as latitudes, sendo maiores próximo ao equador e menores próximo aos polos. (Alibaba Cloud)
- **Precisão Limitada**: Para aplicações que requerem alta precisão, as strings do Geohash podem se tornar longas, o que pode impactar a eficiência.
- **Problemas de Adjacência**: Locais geograficamente próximos podem ter Geohashes significativamente diferentes se estiverem em lados opostos de uma fronteira de célula, complicando buscas por proximidade. (Open Source GIS Data)
- **Limitado a Pontos**: O Geohash é mais adequado para representar pontos. Representar linhas ou polígonos pode ser complexo e menos eficiente. (GIS Stack Exchange)

Aula Avançada: Implementando o Algoritmo Geohash em C e Python

O Geohash é um algoritmo que converte coordenadas geográficas (latitude e longitude) em uma string alfanumérica compacta. Essa string representa uma célula retangular em uma grade hierárquica que cobre o globo. A cada caractere adicionado à string, a célula é subdividida, aumentando a precisão da localização.

Etapas do Algoritmo:

1. **Intervalos Iniciais**: Latitude varia de -90 a +90 graus; longitude de -180 a +180 graus.
2. **Divisão Recursiva**: Para cada bit, o intervalo atual é dividido ao meio. Se a coordenada estiver na metade superior, adiciona-se '1'; caso contrário, '0'.(Wikipedia, Stackademic)
3. **Intercalação de Bits**: Os bits de longitude e latitude são intercalados para formar uma sequência binária.

4. **Codificação Base32**: A sequência binária é agrupada em blocos de 5 bits e convertida em caracteres usando um alfabeto base32 específico, excluindo caracteres ambíguos como 'a', 'i', 'l' e 'o'.

Essa abordagem permite que locais próximos compartilhem prefixos semelhantes em suas representações Geohash, facilitando buscas por proximidade.

Implementação em Python

Python oferece bibliotecas que simplificam a codificação e decodificação de Geohashes.(GitHub)

Exemplo de Codificação:

```
import geohash

latitude = 37.421542
longitude = -122.085589

# Codifica as coordenadas em um Geohash com precisão de 12 caracteres
geohash_code = geohash.encode(latitude, longitude, precision=12)
print(f"Geohash: {geohash_code}")
```

Exemplo de Decodificação:

```
# Decodifica o Geohash de volta para coordenadas
decoded_lat, decoded_lon = geohash.decode(geohash_code)
print(f"Latitude: {decoded_lat}, Longitude: {decoded_lon}")
```

Essas funções permitem converter facilmente entre coordenadas geográficas e Geohashes.

Implementação em C

Para linguagens de baixo nível como C, é necessário implementar o algoritmo manualmente ou utilizar bibliotecas existentes, como a `libgeohash`.(GitHub)

Exemplo de Codificação com `libgeohash`:

```
#include <stdio.h>
#include "geohash.h"

int main() {
    double latitude = 37.421542;
    double longitude = -122.085589;
    int precision = 12;
    char geohash_code[precision + 1];
```

```
    geohash_encode(latitude, longitude, precision, geohash_code);
    printf("Geohash: %s\n", geohash_code);

    return 0;
}
```

Exemplo de Decodificação com `libgeohash`:

```
#include <stdio.h>
#include "geohash.h"

int main() {
    const char* geohash_code = "9q8yyzzzzzzz";
    double latitude, longitude;

    geohash_decode(geohash_code, &latitude, &longitude);
    printf("Latitude: %f, Longitude: %f\n", latitude, longitude);

    return 0;
}
```

Esses exemplos utilizam funções da biblioteca `libgeohash`, que devem ser previamente compiladas e incluídas no projeto.(GitHub)

Estrutura da Codificação Geohash sem biblioteca base

Passos do Algoritmo

1. Definir os intervalos iniciais:
    - Latitude: [-90, 90]
    - Longitude: [-180, 180]
2. Intercalar bits de longitude e latitude (começando por longitude).
3. Converter os bits agrupados de 5 em 5 para a codificação Base32 específica do Geohash:

```
"0123456789bcdefghjkmnpqrstuvwxyz"
```

🐍 Implementação em Python

```
# Base32 Geohash
BASE32 = '0123456789bcdefghjkmnpqrstuvwxyz'
```

```python
def geohash_encode(latitude, longitude, precision=12):
    lat_interval = [-90.0, 90.0]
    lon_interval = [-180.0, 180.0]
    bits = []
    even = True

    while len(bits) < precision * 5:
        if even:
            mid = (lon_interval[0] + lon_interval[1]) / 2
            if longitude > mid:
                bits.append(1)
                lon_interval[0] = mid
            else:
                bits.append(0)
                lon_interval[1] = mid
        else:
            mid = (lat_interval[0] + lat_interval[1]) / 2
            if latitude > mid:
                bits.append(1)
                lat_interval[0] = mid
            else:
                bits.append(0)
                lat_interval[1] = mid
        even = not even

    # Agrupar de 5 em 5 bits e converter para base32
    geohash = ''
    for i in range(0, len(bits), 5):
        chunk = bits[i:i+5]
        index = int(''.join(map(str, chunk)), 2)
        geohash += BASE32[index]

    return geohash

# Teste
print(geohash_encode(37.421542, -122.085589, precision=12))
```

⚙️ Implementação em C

> Por simplicidade, vamos codificar até 12 caracteres. Evitamos usar bibliotecas externas.

```c
#include <stdio.h>
#include <string.h>
#include <math.h>

const char BASE32[] = "0123456789bcdefghjkmnpqrstuvwxyz";
```

```c
void encode_geohash(double latitude, double longitude, int precision,
char *geohash) {
    double lat_interval[2] = {-90.0, 90.0};
    double lon_interval[2] = {-180.0, 180.0};

    int is_even = 1;
    int bit = 0, ch = 0;
    int geohash_index = 0;
    geohash[0] = '\0';

    while (geohash_index < precision) {
        double mid;
        if (is_even) {
            mid = (lon_interval[0] + lon_interval[1]) / 2;
            if (longitude > mid) {
                ch |= (1 << (4 - bit));
                lon_interval[0] = mid;
            } else {
                lon_interval[1] = mid;
            }
        } else {
            mid = (lat_interval[0] + lat_interval[1]) / 2;
            if (latitude > mid) {
                ch |= (1 << (4 - bit));
                lat_interval[0] = mid;
            } else {
                lat_interval[1] = mid;
            }
        }

        is_even = !is_even;

        if (++bit == 5) {
            geohash[geohash_index++] = BASE32[ch];
            bit = 0;
            ch = 0;
        }
    }

    geohash[geohash_index] = '\0';
}

// Teste
int main() {
    char geohash[13]; // 12 chars + null terminator
    encode_geohash(37.421542, -122.085589, 12, geohash);
    printf("Geohash: %s\n", geohash);
    return 0;
}
```

✅ Explicações Finais

- **Precision**: o número de caracteres da string final. Cada caractere representa 5 bits → maior precisão = mais subdivisões.
- **Intercalação**: a ordem alternada entre longitude e latitude preserva a continuidade espacial.
- **Base32**: padrão específico com caracteres que evitam confusão visual (sem 'l', 'o', etc).

Essa implementação mostra como o Geohash é baseado em divisões binárias e em encoding eficiente. Usá-lo sem bibliotecas externas é ideal para sistemas embarcados, bancos de dados sem tipos espaciais e fins educacionais.

Se quiser, posso ajudar a implementar também a **decodificação** de um Geohash para latitude/longitude. Deseja isso?

Conclusão

O Geohash é uma ferramenta poderosa para a codificação e indexação de dados geoespaciais, oferecendo uma abordagem simples e eficiente para representar locais na Terra. Sua estrutura hierárquica e a capacidade de ajustar a precisão o tornam adequado para diversas aplicações, desde sistemas de navegação até bancos de dados geoespaciais.

No entanto, é importante estar ciente de suas limitações, especialmente em aplicações que exigem alta precisão ou envolvem representações complexas de formas geográficas. Em tais casos, outras estruturas de dados, como R-trees ou sistemas baseados em curvas de preenchimento de espaço diferentes, podem ser mais apropriadas.(GIS Stack Exchange)

Em resumo, o Geohash é uma solução elegante para muitos desafios relacionados à geolocalização, e seu entendimento é essencial para profissionais e estudantes que trabalham com dados espaciais.
