Linked Lists in C

Muralidhara V N

IIIT Bangalore

July 18, 2019

Linked List

```
struct node{
        int data;
        struct node *next;};
```

Linked List Traversal

```
bool search (struct node *head, int x){
        while (head!=NULL){
                if(head→ data == x) return true;
                head=head→ next;
        }
return false;}
```

Add a node in the beginning of a list

```
addatbeg(struct node **head, int key){
        struct node *temp;
        temp =malloc(sizeof(struct node));
        temp→data=key;
        temp→next=*head;
        *head=temp;
}
```

Delete a node in the beginning of a list

```
deleteatbeg(struct node **head){
    struct node *temp;
    if(*head!=NULL){
        temp=*head;
        *head=temp→next;
        free(temp);}
}
```

Reverse a linked list

Reverse a linked list

```
reverse(struct node **head){
        struct node *p=NULL,*c=*head,*n;
        while (c!=NULL){
                n=c→ next;
                c→ next =p;
                p=c;
                c=n;
        }
        *head=p;
}
```

Doubly Linked List

```
struct node{
        int data;
        struct node *next, *prev;};
```

Dynamic Data Set

Dynamic Data Set

Hash Tables

$$h : U \rightarrow \{0, 1, 2, \ldots m\}$$

Hash functions

$$h : U \to \{0, 1, 2, \ldots m\}$$

The division method: $h(k) = k \bmod m$.

Hash functions

$$h : U \rightarrow \{0, 1, 2, \ldots m\}$$

The division method: $h(k) = k \bmod m$.
The multiplicative method: $h(k) = \lfloor m(kA \bmod 1) \rfloor$., where $0 < A < 1$

Collision resolution

1. By Chaining

Collision resolution

1. By Chaining
2. Open addressing
   - Linear Probing: $h(k, i) = h(k) + i \mod m$.

Collision resolution

1. By Chaining
2. Open addressing
   - Linear Probing: $h(k, i) = h(k) + i \mod m$.
   - Quadratic probing : $h(k, i) = h(k) + c_1 i + c_2 i^2 \mod m$.

Collision resolution

1. By Chaining
2. Open addressing
   - Linear Probing: $h(k, i) = h(k) + i \mod m$.
   - Quadratic probing : $h(k, i) = h(k) + c_1 i + c_2 i^2 \mod m$.
   - Double hashing : $h(k, i) = h_1(k) + i h_2(k) \mod m$.

Univesal hashing

A collection H of hash functions, $h : U \rightarrow \{0, 1, 2, \ldots m\}$ is said to be **Universal** if for each pair of distinct keys $x, y \in U$, the number of hash functions $h \in H$ for which $h(x) = h(y)$ is atmost H/m.

Univesal hashing

A collection H of hash functions, $h : U \rightarrow \{0, 1, 2, \ldots m\}$ is said to be **Universal** if for each pair of distinct keys $x, y \in U$, the number of hash functions $h \in H$ for which $h(x) = h(y)$ is atmost H/m.

.

In other words, with a hash fucntion randomly chosen from H , the chances that $h(x) = h(y)$ is less than $1/m$.

Univesal hashing

A collection H of hash functions, $h : U \rightarrow \{0, 1, 2, \ldots m\}$ is said to be **Universal** if for each pair of distinct keys $x, y \in U$, the number of hash functions $h \in H$ for which $h(x) = h(y)$ is atmost H/m.

.

In other words, with a hash fucntion randomly chosen from H , the chances that $h(x) = h(y)$ is less than $1/m$.

.

If the size of the table is n^2, then probability that there will be no collision is less than $1/2$.

Perfect hashing

no collision.