M S Ramaiah Institute Of Technology

(Autonomous Institute, Affiliated to VTU)

VISUALIZING MOLECULAR STRUCTURE USING MODERN DATA STRUCTURES

Submitted in partial fulfilment of the CIE for the subject
Data Structures(IS333)

By:
Roopak S(1MS15IS095)
Shreehari N W(1MS15IS114)

Under the guidance of
Rajeshwari S B
Assistant Professor
Department of ISE,MSRIT
Bangalore-560054

Contents

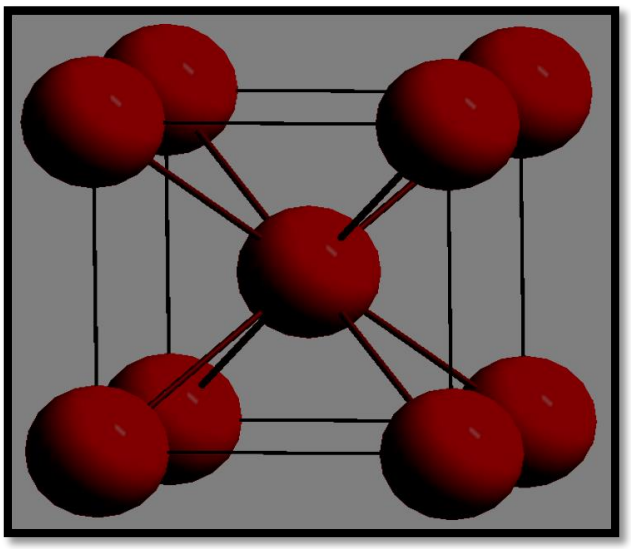
Abstract

A molecule is comprised of atoms which are arranged in a symmetric recurring fashion. In chemistry it in necessary to understand this structure to create new alloys, compounds and substances. Even one nano gram of a substance can contain millions of molecules and each one will behave differently during a chemical reaction is not possible to humanely understand every change. Hence to understand how each molecule behaves during a reaction we need powerful computers to simulate the responses of every molecules

In this project we try to simulate such analytics on a very small scale using modern data structures.

Introduction

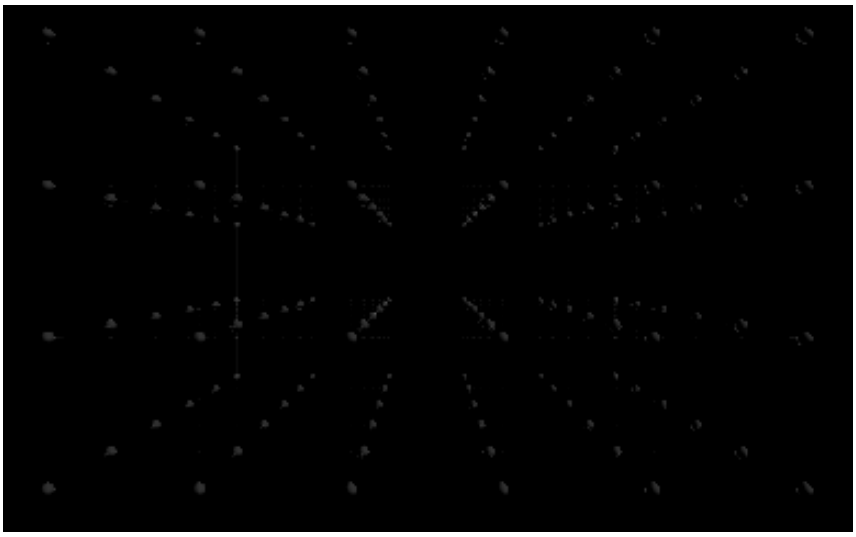
The main aim of this project is to create the lattice structure of an element. The smallest unit of the lattice structure is called a unit cell. There are about 9 different types of unit cells and we have chosen to replicate the body centered unit cell.



Body centered cubic unit cell

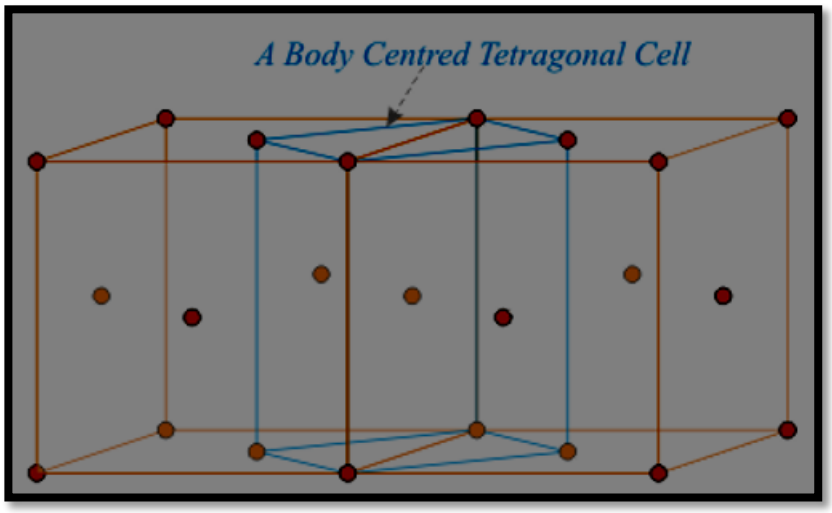
This kind of a unit cell consists of one atom at the centre of the unit cell surrounded by 8 others atoms which are oriented like the corners of a square.

In order to from an element millions of such unit cells combine to from the lattice



Since we are dealing with very low ssytem capabilities we will on be creating a lattice in one dimensional

A pictorial view of what will be created in memory is given below



Source Code

```
#include <stdio.h>
#include<stdlib.h>
#include<malloc.h>
void insert();
struct node
{
   int info;
};
int count=0,k=1;
struct unitcell
{
   int ucinfo;
   struct unitcell * left;
   struct unitcell * right;
   struct node * n[8];
};

struct unitcell*start=NULL;
struct unitcell*temp;
struct unitcell *nn;

void create(int item,int item1)
{
   int i;
   nn=((struct unitcell*)(malloc(sizeof(struct unitcell))));
   nn->ucinfo=item;
   nn->left=NULL;
   nn->right=NULL;
   if(start==NULL)
   {
                start=nn;
   }
   else
   {
      insert(item1);return;
   }
   for(i=0;i<8;i++)
   {
                nn->n[i]=((struct node *)(malloc(sizeof(struct node))));
      nn->n[i]->info=item1;
   }
}

void insert(int item1)
{
   int i;
   if(count%2==0)
   {
      for(i=1;i<8;i=i+2)
         {
```

```
                                nn->n[i]=((struct node *)(malloc(sizeof(struct node))));
            nn->n[i]->info=item1;
          }

                temp=start;
                while(temp->right!=NULL)
                {
                        temp=temp->right;
                }
                temp->right=nn;
                nn->left=temp;
                printf("inserted right\n");count++;return;
          }
          if(count%2==1)
          {
                for(i=0;i<8;i=i+2)
       {
                        nn->n[i]=((struct node *)(malloc(sizeof(struct node))));
          nn->n[i]->info=item1;
       }

                temp=start;
                while(temp->left!=NULL)
                {
                        temp=temp->left;
                }
                temp->left=nn;
                nn->right=temp;
                printf("inserted left\n");count++;
          }
}


void display()
{
   int i;int j=1;
   struct unitcell*temp1=start;
   while(temp1->left!=NULL)
   {
      temp1=temp1->left;
   }
   while(temp1!=NULL)
   {
      printf("%d\t",temp1->ucinfo);
      if(temp1==start)
      {
           for(i=0;i<8;i++)
        {
           printf("%d\t",temp1->n[i]->info);
        }
        j=0;
       }
```

```
        else if(j!=0)
        {
           for(i=0;i<8;i=i+2)
           {
              printf("%d\t",temp1->n[i]->info);
           }
        }
        else
        {
           for(i=1;i<8;i=i+2)
           {
              printf("%d\t",temp1->n[i]->info);
           }
        }
        printf("\n");
        temp1=temp1->right;
     }
}

void display1()
{
   int i;int j=1;
   struct unitcell*temp1=start;
   while(temp1->left!=NULL)
   {
      temp1=temp1->left;
   }
   struct unitcell*k5=temp1;
   int k=0;
   for(;k<8;)
   {
                  temp1=k5;
      if(k==4)
      {
                        for(;temp1!=NULL;)
         {
                              printf("  %d  ",temp1->ucinfo);
                              temp1=temp1->right;
         }
                  printf("\n");
                  temp1=k5;
      }

      for(;temp1!=start;)
      {
         printf("%d   ",temp1->n[k]->info);
         temp1=temp1->right;
                        if(temp1==start)
                        {
                              printf("%d   ",temp1->n[k]->info);
                        }
      }
```

```
        k++;

        for(;temp1!=NULL;)
        {
           printf("%d   ",temp1->n[k]->info);
           temp1=temp1->right;
        }
        k++;
        printf("\n");
     }
}

void search(int key)
{
    int i;int j=1;
    temp=start;
    while(temp->left!=NULL)
        temp=temp->left;

           while(temp!=NULL)
    {
        if(temp==start)
        {
              for(i=0;i<8;i++)
           {
              if(temp->n[i]->info==key)
              {
                 printf("%d\t",temp->ucinfo);
                 for(j=0;j<8;j++)
                    printf("%d\t",temp->n[j]->info);
                 return;
              }

           }
           k=0;
        }
        else if(k!=0)
        {
           for(i=0;i<8;i=i+2)
           {

              if(temp->n[i]->info==key)
              {
                 printf("%d\t",temp->ucinfo);
                 for(j=0;j<8;j=j+2)
                    printf("%d\t",temp->n[j]->info);
                 return;
              }
           }
        }
        else
        {
```

```
            for(i=1;i<8;i=i+2)
            {
              if(temp->n[i]->info==key)
              {
                  printf("%d\t",temp->ucinfo);
                  for(j=1;j<8;j=j+2)
                     printf("%d\t",temp->n[j]->info);
                  return;
               }
            }
        }
        temp=temp->right;
    }
}


void delete(int key)
{
    int i;
    search(key);
    if(temp==start||temp==NULL)
    {
        printf("invalid choice\n");return;
    }
    if(k==1)
    {
        for(i=0;i<8;i=i+2)
        {
            free(temp->n[i]);
        }
        struct unitcell *k1,*k2;
        k1=temp->right;
        k2=temp->left;
        k1->left=k2;
        k2->right=k1;
        free(temp);
    }
    if(k==0)
    {
        for(i=1;i<8;i=i+2)
        {
            free(temp->n[i]);
        }
        struct unitcell *k1,*k2;
        k1=temp->right;
        k2=temp->left;
        k1->left=k2;
        k2->right=k1;
        free(temp);
    }
}
```

```
void impact(int imp)
{
   int imp1=imp,t;
   struct unitcell*s1=start;
   while(s1!=NULL)
   {
      t=imp/1.1732;
      if(t>1)
      {
         s1->n[1]->info-=t;
         s1->n[3]->info-=t;
         s1->n[5]->info-=t;
         s1->n[7]->info-=t;
         imp=t;
      }
      else
      {
         break;
      }
      s1=s1->right;
   }
   s1=start;
   imp=imp1;
   while(s1!=NULL)
   {
      t=imp/1.1732;
      if(t>1)
      {
         s1->n[0]->info-=t;
         s1->n[2]->info-=t;
         s1->n[4]->info-=t;
         s1->n[6]->info-=t;imp=t;
      }
      else
      {
         break;
      }
      s1=s1->left;
   }
         return;
}

//////////////////MAIN MODULE//////////////////
int main()
{
   while(1)
   {
      int ch,cval,val_search,val;
      printf("\nMENU\n");
      printf("1.INSERT\n2.SEARCH\n3.DELETE\n4.DISPLAY AS
STRUCTURE\n5.DISPLAY\n6.    IMPACT\n7.EXIT\n");
      scanf("%d",&ch);
```

```
        switch(ch)
        {
           case 1:printf("enter the value of core and its vertices\n");
              scanf("%d%d",&cval,&val);
              create(cval,val);
              break;
           case 2:printf("enter the value to search\n");
              scanf("%d",&val_search);
              search(val_search);
              if(temp==NULL)
                 printf("Not found\n");
              break;
           case 3:printf("enter the value to delete\n");
              scanf("%d",&val_search);
              delete(val_search);
              break;
           case 4:display1();break;
           case 5:display();break;
           case 6:printf("enter the value to IMPACT\n");
              scanf("%d",&val_search);
              impact(val_search);
              break;
           case 7:exit(0);break;
           default:printf("ivalid choice");

        }

     }
     return 0;
}
```

SUMMARY

struct node -
This structure defines every node that will be linked to a core

struct unitcell -
This structure defines the characteristics of the core of each unit cell.It contains eight pointers that refer to the eight corners of the unit cell(every node structure)

create(core threshold, node threshold)
This is the core function of the program which creates the unit cells with the given threshold values for core an nodes.It falles the insert function to link the unit cell to the chain

Insert(threshold value) -
This function adds a unit cell to the chain of unitcells.Inorder to create a balances chained insertion occurs alternatively to the left and right of the centre

Display() -
this fuction displays the threshold values of all he nodes and cores in the chain in a table format for detailed analysis of the chain.

Display1() -
this function displays the unit cell chain in a graphical format. This is useful to understand logically the chain structure

Search(threshold)
This function searches the chain for all nodes having a threshold value lower than that specified

Delete(threshold) -
This function is used to delete any unit cell having nodes with thresholds lower than that specified.

Impact(value) -
This is the function that adds purpose to the program.
To this function we pass a impact value. The computer then subjects the core of the chain to the given impact value and recursively simulates the propagation of impact to the neighboring unit cells. The amount of impact passed on to the neighbors is given by a formula that will be characteristic of every element being recreated.

Acknowledgments

I would like to express my special thanks of gratitude to my teacher Ms.Rajeswari S B who gave us the opportunity to do this wonderful project by teaching us the various concepts essential to this project's existence, which also helped us do a lot of research and learn new topics related to data structures and C programming.
Secondly we would also like to thank our parents and friends who helped us finalize this project within the limited time frame.