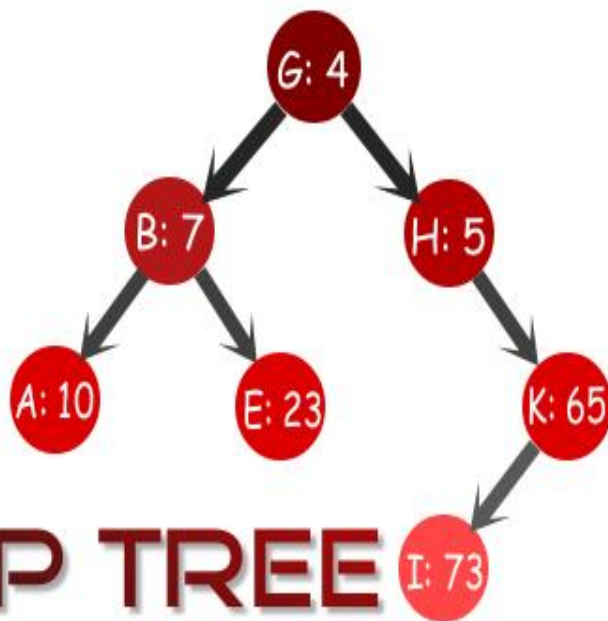




مشروع الخوارزميات وبنى المعطيات [2]



TREAP TREE

اشراف المهندس:

سوسن حسن

اعداد الطلاب:

عبد الرحمن الحوراني

محمد عيد التل

وائل حموش

محمد الخياط

يعتبر نظام الأشجار نظاماً مهماً جداً في عمليات عدة منها عمليات التخزين في الذاكرة وأيضاً البحث ... الخ

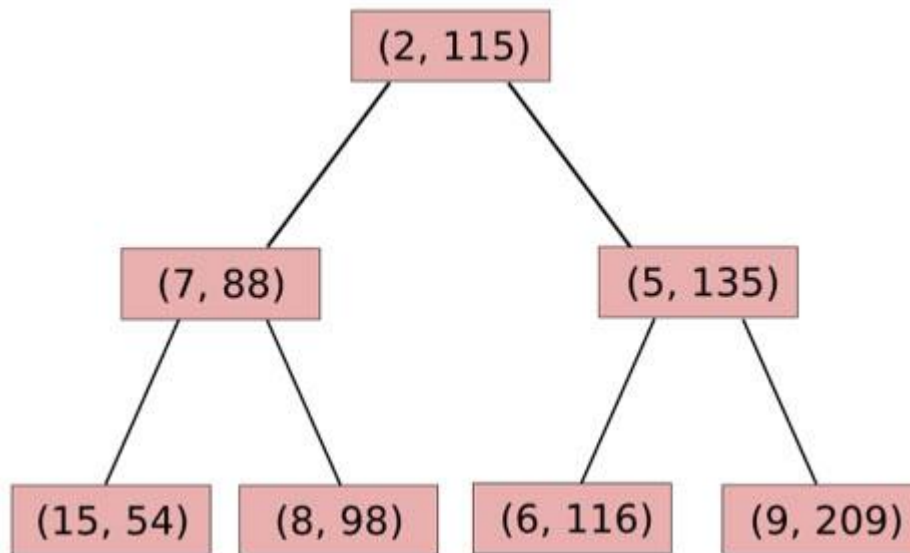
يوجد أشجار عدة مثل شجرة البحث الثنائي BST وشجرة الكومة Heap Tree وايضاً شجرة الـ AVL

سنتحدث في هذا التقرير عن نوع مهم من الأشجار وهي شجرة الـ Treap وهي شجرة تحقق خواص شجرة البحث الثنائي BST وشجرة الـ AVL تكمن أهمية هذه الشجرة في خفض ارتفاع شجرة البحث الثنائية. تأخذ كل عقدة من هذه الشجرة قيمتين وهما الـ key والـ value.

القيمة **key** تحقق شروط الشجرة الثنائية BST

والقيمة **value** تحقق شروط شجرة الـ AVL

يمثل الشكل التالي نموذجاً عن شجرة الـ Treap:



البنية المطلوبة:

Struct TreeNode وهي عقدة من الشجرة وتحوي على:

Key : متحول من نمط int سيتم ترتيب الشجرة من خلاله على اساس انها شجرة بحث ثنائية وتمثل بيانات العقدة ويتم الترتيب حسبها

value: متحول من نمط int سيتم ترتيب الشجرة من خلاله على اساس انها شجرة كومة (heap tree) من النوع min heap وتمثل ال priority

TreeNode *left: مؤشر من نمط العقدة يُوْشِر على الابن اليساري.

TreeNode*right: مؤشر من نمط العقدة يُوْشِر على الابن اليميني .

class tree وهي كامل الشجرة والتي تحوي على العقدة وعلى التوابع الخاصة بالشجرة

TreeNode*root وهو يمثل العقدة الاولى العقدة الاب لكل العقد

Tree() يمثل الباني constructor

~Tree() يمثل الهادم destructor

"السبيل الوحيد للقيام بعمل عظيم هو ان تحب ما تعمل"

ستيف جوبز

جدول التحليل الأولي

اسم التابع	bool is Empty
المهمة	يقوم بالتحقق فيما إذا كانت الشجرة فارغة ام لا
الخوارزمية	يتحقق إذا كان الجذر (root) فارغ ام لا

جدول التحليل المفصل

اسم التابع	TreeNode *SearchNode()
المهمة	البحث عن عقدة معينة ضمن شجرة وذلك عن طريق الـ key
توصيف المتحولات	يأخذ العقدة الـ root ومفتاح key ل يبحث عن عقدة تحوي هذا المفتاح
الخوارزمية	نأخذ العقدة الـ root ونبحث عن الـ key بالمقارنة معها اذا كانت العقدة المرادة اكبر من الـ root نبحث في الـ subtree اليميني واذا كانت اصغر من الـ root نبحث في الـ subtree اليساري وتتم تكرار هذه العملية عودياً عن كل عقدة

```

109   if (Node == NULL) return Node;
110   else {
111       if (Key < Node->Key)
112           SearchNode (Node->left, Key);
113       if (Key > Node->Key)
114           SearchNode (Node->right, Key);
115       else return(clone (Node));
116   }

```

عندما يصل الى عقدة فارغة

حالة الانتهاء

```

109   if (Node == NULL) return Node;

```

او عندما يصل الى العقدة المطلوبة

```

115       else return(clone (Node));

```

جدول التحليل الأولي

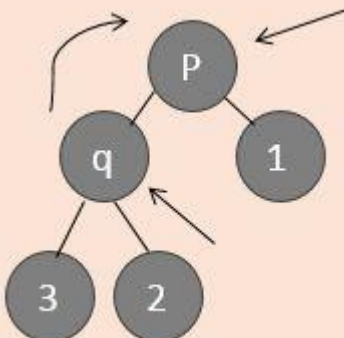
clone

اسم التابع

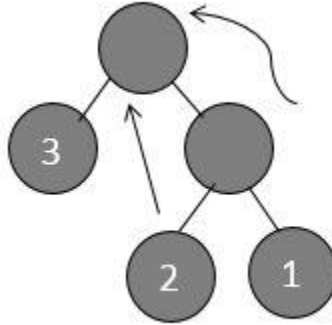
يقوم بنسخ عقدة ما وارجاع النسخة

المهمة

جدول التحليل المفصل

اسم التابع	Rotation
	Left –to- Right
المهمة	يقوم التابع بتدوير العقدة عند حدوث اختلال في ال priority اي ال value
توصيف المتحولات	نأخذ العقدة المراد التدوير عندها
الخوارزمية	نسمي العقدة اليسارية للعقدة المرادة p ب q ونأخذ العقدة اليمينية ل q لتصبح عقدة يسارية ل p ونجعل العقدة p هي عقدة يمينية ل q ونقوم بتسمية p ب q
	

Right-To-Left



جدول التحليل المفصل

اسم التابع	Insert
المهمة	يقوم التابع بإنشاء عقدة ووضعها في المكان المناسب حسب الkey ويختبر ال priority الخاص بها ليقوم بتعديل مكانه في حال كانت ال priority الخاصة به ليست في مكانها الصحيح
توصيف المتحولات	نأخذ معلومات جميع العناصر المراد وضعها في العقدة والعقدة ال root
الخوارزمية	نقوم بالبحث عن المكان المناسب للعقدة عن طريق ال key حيث نقارنه مع العقدة الحالية (التي تكون root في المرة الاولى) فيما اذا كانت اكبر منها نبحث في ال subtree اليمينية او اصغر منها نبحث

في subtree اليسارية وعندما نصل الى مكان null عندها يكون هذا المكان هو المكان المناسب وعندها نقوم بإنشاء العقدة بالمعطيات المرادة وندخلها لهذا المكان

```
141     if (temp == NULL) {
142         temp = new TreeNode();
143         temp = clone(newNode);
144     }
```

حيث NewNode هي العقدة التي أنشأناها بالمعطيات المطلوبة والتي قمنا بإدخالها في المكان المناسب وبعد ان ندخلها عندها نكون قد بدأنا بالمرحلة الثانية وهي اختبار الاولوية ونلاحظ وجود حالتين الحالة الاولى: عندما نكون قد أدخلنا العقدة في الـ subtree اليسار نقارن بعدها في حال كانت priority العقدة الحالية أكبر من priority العقدة التي قمنا بإضافتها في اليسار عندها نقوم بالتدوير عند العقدة الحالية بتدوير left_to_right الحالة الثانية: تمثل العكس تماماً

عندما نصل الى عقدة null وعندها نقوم بإضافة العقدة الجديدة فيها

حالة الانتهاء

```
141     if (temp == NULL) {
142         temp = new TreeNode();
143         temp = clone(newNode);
144     }
```


جدول التحليل المفصل

اسم التابع	Delete
المهمة	يقوم التابع بحذف عقدة محددة من الشجرة مع مراعاة بقاء الشجرة متوازنة من حيث ال key وال priority
توصيف المتحولات	نأخذ العقدة ال root للشجرة وقيمة x هي قيمة ال key العقدة التي نبحث عنها
الخوارزمية	<p>نقوم اولا بعملية ال search مشابهة لتابع ال search السابق وعندما نصل للعقدة المطلوبة نقوم بالتالي [(بشكل عام) : عندما نجد العقدة نقوم بالتدوير عندها حتى تصبح العقدة لا تحوي ابناء ليصبح حذفها اسهل]</p> <p>1- إذا كانت العقدة لا تحوي ابن يساري ولا ابن يميني عندها نقوم بحذف هذه العقدة</p> <pre> 204 if (p->left == NULL && p->right == NULL) { 205 delete p; 206 p = NULL; 207 return; 208 }</pre> <p>2- إذا كان الابن اليساري null والابن اليميني للعقدة ليس null للعقدة المراد حذفها نقوم بتدوير ال Right-to-Left ونستدعي الحذف عند العقدة بعد التدوير</p>

```

209     if (p->left == NULL && p->right != NULL) {
210         right_to_left(p);
211         del(p->left);
212         return;
213     }

```

3- إذا كان البن اليساري ليس null والبن اليميني null للعقدة المراد

حذفها نقوم بتدوير Left_to_Right ثم نستدعي تابع الحذف من

اجل العقدة بعد التدوير

```

214     if (p->left != NULL && p->right == NULL) {
215         left_to_right(p);
216         del(p->right);
217         return;
218     }

```

4- إذا كان ال priority الابن اليساري للعقدة أصغر من ال priority

الابن اليميني للعقدة نقوم بالتدوير Left_to_Right ونستدعي

التابع من اجل العقدة بعد التدوير

```

219     if (p->left->iValue < p->right->iValue) {
220         left_to_right(p);
221         del(p->right);

```

5- إذا كان ال priority الابن اليساري للعقدة أكبر من ال priority

الابن الايمن للعقدة عندها نقوم بتدوير Right_to_Left ونستدعي

تابع الحذف من اجل العقدة بعد التدوير

```

222         } else {
223             right_to_left(p);
224             del(p->left);
225         }

```

1- حتى نصل الى عقدة null اي لا توجد عقدة تحوي الـ x المراد

حالة الانتهاء

```

194         if (p == NULL) return;

```

2- او عندما نصل للعقدة المطلوبة ونحذفها ونقوم بموازنة الشجرة

للحفاظ على الـ priority

```

204         if (p->left == NULL && p->right == NULL) {
205             delete p;
206             p = NULL;
207             return;
208         }
209         if (p->left == NULL && p->right != NULL) {
210             right_to_left(p);
211             del(p->left);
212             return;
213         }
214         if (p->left != NULL && p->right == NULL) {
215             left_to_right(p);
216             del(p->right);
217             return;
218         }

```

جدول التحليل المفصل

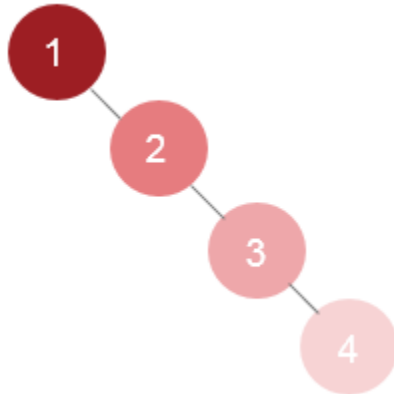
اسم التابع	Splite
المهمة	يقوم بتقسيم الشجرة الى شجرتين بالنسبة للـ key حيث ينتج شجرة أولى تحوي جميع العقد الأكبر من الـ key والشجرة الثانية تحوي جميع العقد الأصغر من الـ key
توصيف المتحولات	يأخذ الـ root الشجرة المراد تقسيمها و root الشجرة المراد التقسيم عليها والـ key المراد التقسيم على اساسه
الخوارزمية	<p>1- نقارن الـ key المراد مع key العقدة الأولى</p> <p>2- إذا كان key العقدة أكبر من مؤشر الشجرة اليمينية نؤشر على هذه العقدة ثم نستدعي التابع نفسه من أجل العقدة اليسارية (حيث انه خمننا انه جميع العقد على يمين العقدة هي أكبر من الـ key التي نقسم بها)</p> <p>3- اما إذا كان الـ key العقدة أصغر من key المراد التقسيم بها نجعل مؤشر الشجرة اليسارية يؤشر عليها ثم نستدعي التابع نفسه من اجل العقدة اليمينية للشجرة الأساسية (حيث اننا ضمننا انه جميع العقد اليسارية للعقدة التي قارننا بها هي أصغر من key التي نقسم بها) ونغير مؤشر الشجرة اليسارية الى مؤشر الابن اليمين لها</p>

4- وعند الوصول NULL في الشجرة الأساسية نقوم بتسكير الشجرتين اليمينية واليسارية بوضع NULL على مؤشراتها لإيقاف العمل وانتهاء التقسيم	
ينتهي التابع عندما نصل الى نهاية الشجرة المراد تقسيمها	حالة الانتهاء

اسم التابع	Join
المهمة	يدمج شجرتين في شجرة أخرى
توصيف المتحولات	يأخذ root الشجرة الأولى و root الشجرة الثانية ليدمجها
الخوارزمية	نقوم بإضافة عناصر الشجرة الثانية والشجرة الأولى الى شجرة أخرى
	<pre> 290 if (b != NULL) { 291 292 TreeNode *c = new(TreeNode) ; 293 c = clone(b) ; 294 Insert(a,c) ; 295 join(a,b->left) ; 296 join(a,b->right) ; 297 }</pre>
حالة الانتهاء	عندما تنتهي الشجرتين المراد إضافتهما

التعقيد:

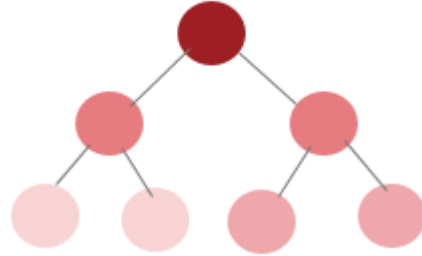
- 1- تابع الـ search: تعقيده هو $O(n)$ وذلك في أسوأ الأحوال في حالة تم إضافة ارقام مرتبة تصاعديا والـ priority لكل منها يزداد مثل:
 $\{(1,1) (2,2) (3,3) (4,4)\}$



- 2- تابع الـ Insert: $O(n) + O(n) = O(2n) \approx O(n)$ لأنه عبارة عن عمليتين هما:

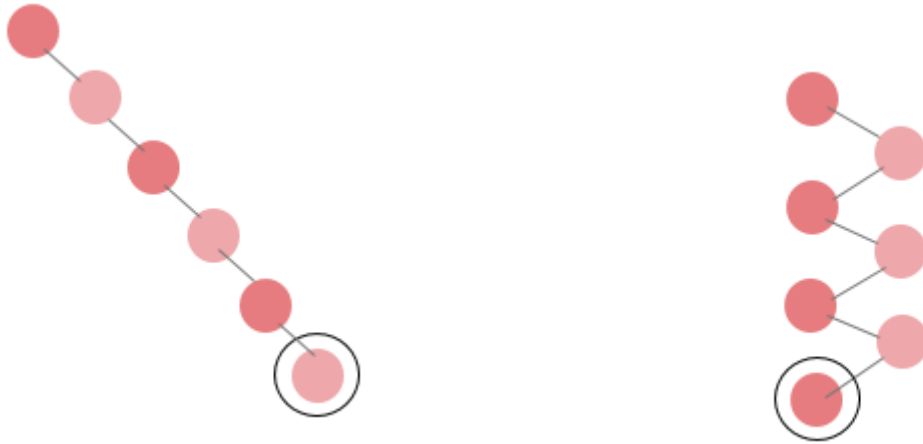
- i. Search: وتعقيدها هو $O(n)$ وعندما يجد المكان المناسب يدخل العقدة ثم:
 - ii. موازنة الشجرة في حال اختلال الـ priority وهي أيضاً $O(n)$ حيث أنه في أسوأ الأحوال يكون الـ priority العقدة المضافة اصغر من priority root وعندها سنقوم بتدوير عند جميع العقد (آبائها) حتى الوصول للـ root
- 3- تابع الـ Delete: $O(n) + 2 \log(n) \approx O(n)$ لأنه عبارة عن:
- i. بحث عن العقدة المطلوبة (Search) أي $O(n)$

II. ثم تدويرها لتصبح ورقة leaf وتعقيدها هو (ارتفاع الشجرة مضروب بـ 2) وهنا الارتفاع الأسوأ هو الارتفاع الكامل المتوازن أي:



أي عندما يكون $h = \log(n)$
اما عندما يكون الارتفاع اكبر من ذلك فيكون التعقيد أقل (أي الحالة الأقل سوء) فالتعقيد بالحالة الأسوأ هو $2\log(n)$

4- تابع الـ Split: تعقيده $O(n)$ لأنه الحالة الأسوأ أن تكون كما في الشكلين الآتيين:

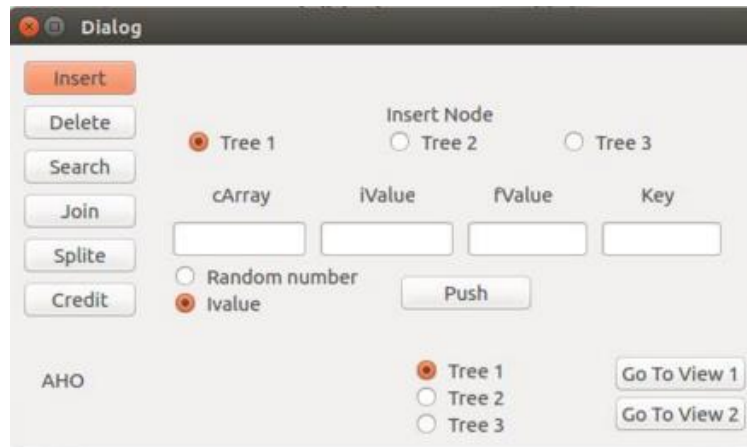


5- تابع الـ join: تعقيده $O(n) \approx O(2n)$ حيث n هي عناصر الشجرة الأولى التي نقوم بإضافة عناصرها الى الشجرة الرئيسية وال n الأخرى عناصر الشجرة الثانية التي نقوم أيضا بإضافتها الى الشجرة الأساسية

القسم الثاني من المشروع ((الواجهات باستخدام الـ QT))

يوجد ثلاثة أشجار رئيسية للإضافة والعمل بها

❖ عند الضغط على Insert: يوجد لدينا Tree 1 | Tree 2 | Tree 3 وتعني بأي شجرة نريد الإضافة



- نقوم بوضع القيم التي نريد إدخالها في عقدة جديدة إلى الشجرة ثم نقوم بالضغط على الـ push لإدخال العقدة

- كما هو موضّح في الشكل: يوجد خيارين هما:

1. Random number: عند الضغط عليه سيصبح مربع الـ iValue

غير متاح لوضع معلومات به وسيتم تنفيذ تابع ادخال عقدة بدون قيمة priority والتي سيتم اضافتها بشكل عشوائي

2. iValue: يصبح مربع الـ iValue متاح وسنقوم بإدخال جميع المعلومات وعندها سيتم تنفيذ تابع ادخال عقدة مع قيمة iValue

- يوجد في الأسفل:

• Tree 1

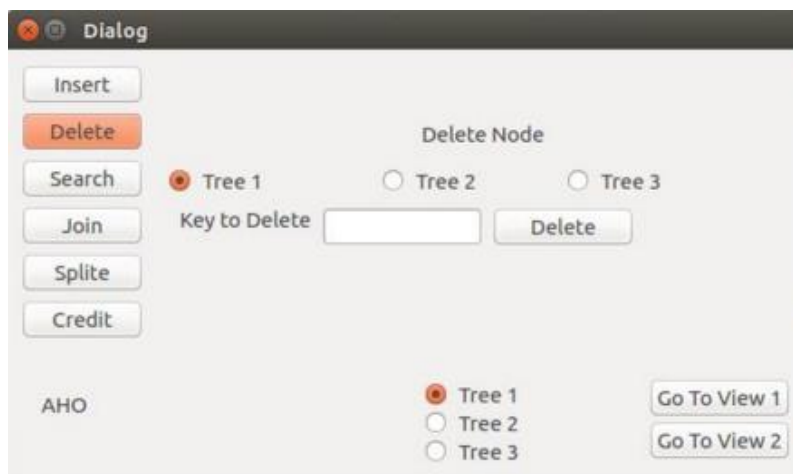
• Tree 2

• Tree 3

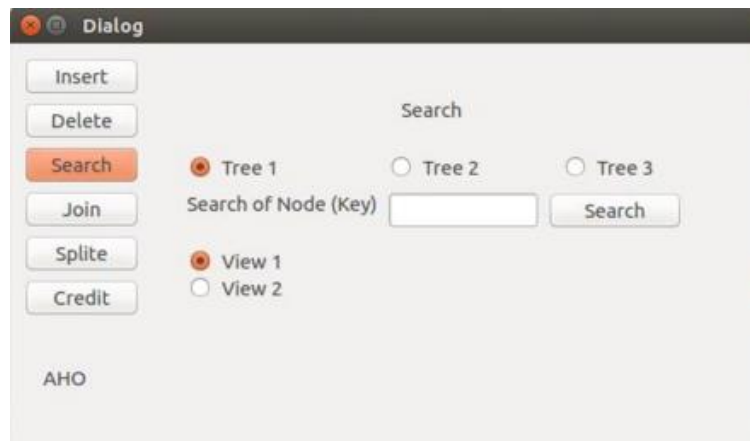
تعني إلى أي شجرة نريد أن نرسم

- Go To view 1: أي نذهب لمشهد مصغر للعقد وذلك ليصبح المشهد أكثر وضوحاً ورؤية جميع العقد فوراً
- Go To view 2: تقوم بإظهار مشهد مُكبر للشجرة مع توضيح ما تحوي من معلومات

❖ عند الضغط على Delete: نضع الـ key الخاص بالعقدة المراد حذفها ونضغط على Delete لحذفها

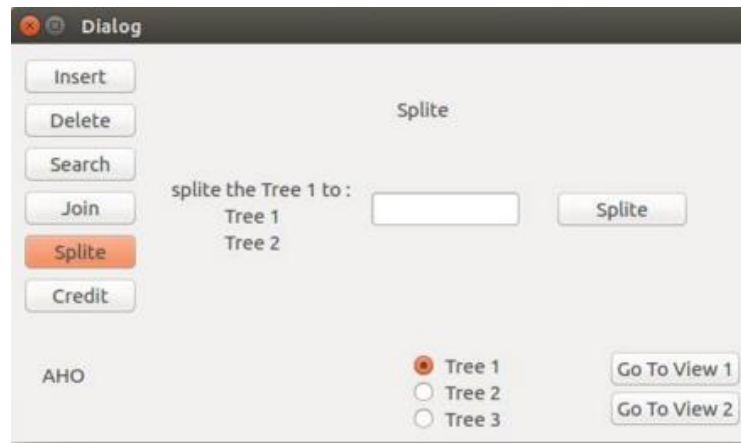


❖ عند الضغط على Search: نضع في المربع الـ key الخاص بالعقدة المراد البحث عنها ونضغط زر الـ Search للبحث وسوف تنتقلنا إلى الشجرة مع تلوين العقدة المراد البحث عنها باللون الأصفر



تنويه: الخيارين view 1 و view 2 لنرى العقدة بشكل مصغر أو مكبر

❖ عند الضغط على Split : نقوم بوضع الـ key الخاص بالعقدة المراد التقسيم عندها ويتم التقسيم عند الضغط على Split



❖ عند الضغط على Join : يقوم بدمج الشجرة الثانية والثالثة في الأساسية



❖ عند الضغط على Credit : يقوم بإظهار أسمائنا ^_^



خوارزميات عامة في الرسم:

سنقوم في الشرح بشكل عام عن الخوارزميات المستخدمة في الرسم والتفصيل موجود في الكود

```
* gotoview() {  
    ui.page ---> show();  
    // إي اظهار الصفحة المراد الذهاب لها  
    scene ---> clear();  
    // أي مسح المشهد لرسم مشهد جديد  
    TreeHeight();  
    // وذلك لموازنة الشجرة وإعطائها متحول Balance للمساعدة في رسم الشجرة  
    ((لا علاقة للـ Balance في الـ Balance الخاص بالـ AVL))  
    DrawEllipse();  
    // لرسم الشجرة
```

```
* draw  
    drawElipse(old_center, center, y, k)  
    old_center = center  
    // يرسم خط من العقدة السابقة إلى الحالية  
  
    Draw(T-->Left, old_center, center—T-->Balance, y+18, k)
```

يرسم العقدة على اليسار

Draw(T-->Right, old_center, center+T-->Balance, y+18, k)

يرسم العقدة على اليمين

```
* drawEllips(TreeNode *T, double old_center, double center, double y, int k)
```

```
QBrush brush(QT::red)
```

```
if(T-->key == k) {
```

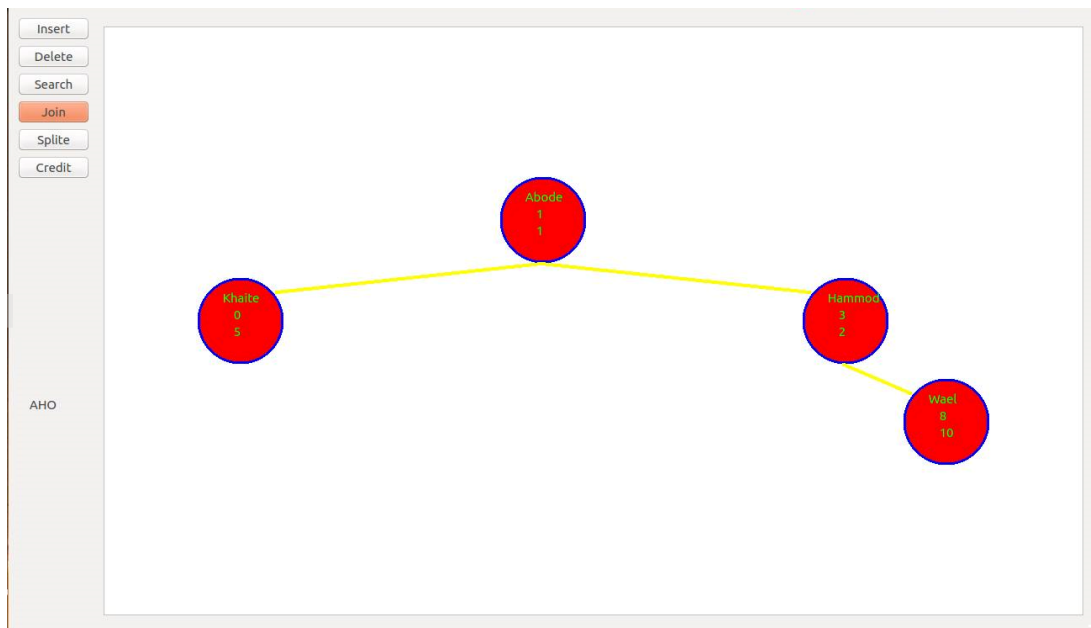
```
brush, setcolor(QT::yellow) }
```

// اذا وجد العقدة المراد البحث عنها يقوم بتلوينها باللون الأصفر

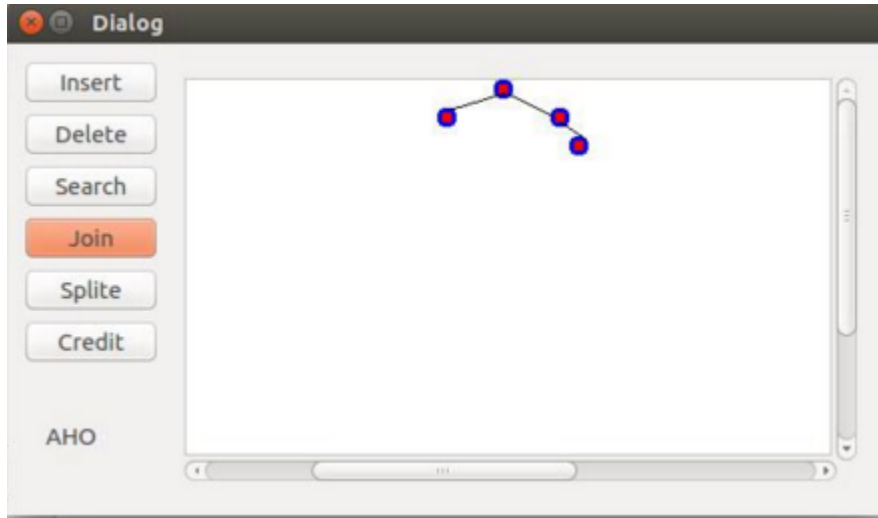
// الباقي موجود في الكود وذلك لإعطاء الاحداثيات اللازمة //

بعض الصور عن التنفيذ:

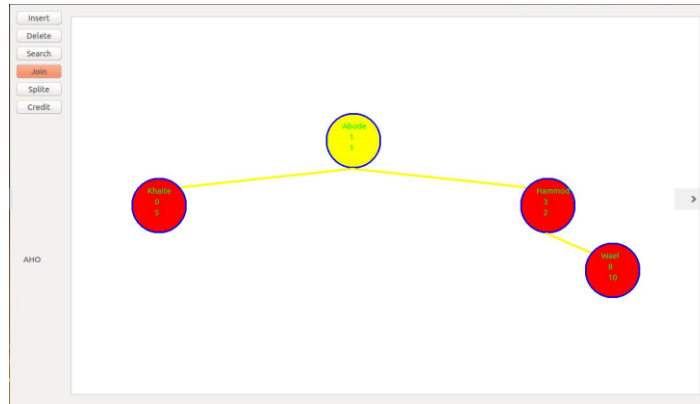
العقد في الشكل المكبر



العقد في الشكل المصغر



العقدة المراد البحث عنها ((ملونة بالأصفر))



هكذا يكون قد انتهى مشروع الخوارزميات (2) بفضل الله
نتمنى أن ينال الاعجاب آملين بعلامات ممتازة
شاكرين جهودك وتعبك معنا خلال هذا الفصل