**Peer-to-peer Protocols and Applications**

**Homework #1 Chord模擬程式**

**請將數據圖及程式寫在一個word檔案，上傳至[DEL]**

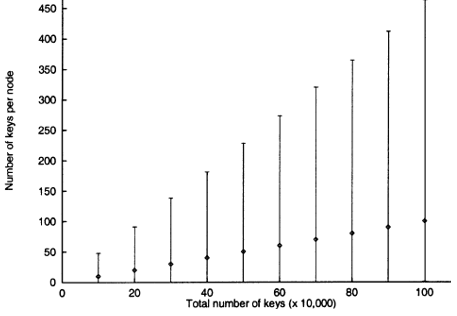
請撰寫模擬程式來計算Chord系統中(每個ID由32 bit表示)，每一個節點所需負責之key值個數，以及搜尋時所需之hop數。系統參數假設如下：

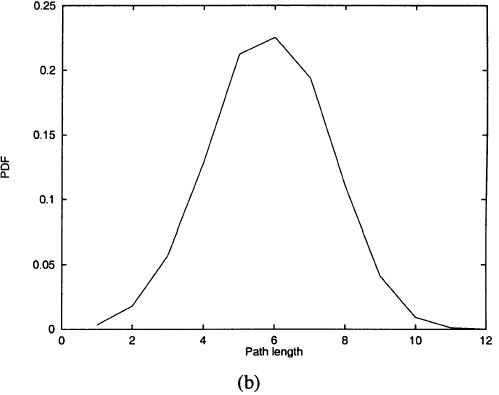
* n = peer 個數 = 10,000
* numK = 所有檔案key值個數 (10\*n, 20\*n, …, 100\*n)

請繪製下列點線圖:

1. 假設*xi*代表node *i*所負責保管的key值總數，請繪出這些*xi*中的最小值、最大值，及中位數(median) vs. numK (=10\*n, 20\*n, …, 100\*n)
2. 執行 100萬次搜尋，繪出所需hops所佔比例 vs. number of hops (from 0 to 15)

圖形類似如下:





模擬程式結構流程大致如下(使用single thread即可，不需要multi-thread):

int n = 10000;

main ( ) {

createTopology( ); // 每個node產生node ID, 建立前後關係以及finger table

mapKeysToPeers( ); // 將檔案擺放到node，計算node所需負責之key值個數

computeSearchHops( ); // 執行 100萬次搜尋，計算所需hop數之分佈

}

createTopology ( ) {

for (int i =0; i<n; i++)

node i隨機產生一個32-bit ID; //不可和已經產生的ID值重複

for (int i =0; i<n; i++)

node i紀錄他的predecessor以及successor;

for (int i =0; i<n; i++)

node i產生finger table;

}

mapKeysToPeers( ) {

int count[n];

for (int i = 10; i<= 100; i = i + 10) {

把count[ ] 全部初始化為0;

for (int j = 0; j<i\*n; j++) { // 產生共 i\*n 個檔案key值，不需要檢查是否重複產生過

int k = 隨機產生的32-bit key值;

假設k值應該由node x保管，則 count[x] ++;

}

印出count[ ] 裡的最小值、最大值，中位數;

}

}

computeSearchHops( ) {

int hops[32] = {0}; // 把hops[ ] 全部初始化為0;

for (int i = 0; i<n; i++) {

for (int j = 0; j<100; j++) { // 每個node *i*做100次搜尋

int k = 隨機產生的32-bit key值;

利用各個node的finger table做routing從i走到負責保管k值的node (假設為 x);

假設走了d hops，則 hops[d] ++;

}

}

印出hop[0] …. hop[31] 的比例值;

}