アルゴリズムとデータ構造　2005 年度冬学期　期末試験

問1　ハッシュ表に直接データを挿入するクローズドハッシュ法について以下の問いに答えよ。ただし、再ハッシュでは残りのバケットが等確率で選ばれるものとする。また、表のうち要素の埋まっているバケットの割合をαとする。計算の過程も簡潔に記すこと。(各 5 点)

1) 新しい要素を挿入する際に調べる平均バケット数を求めよ。
2) すでに格納されている要素を削除する際に調べる平均バケット数を求めよ。

問2　次の再帰方程式を解き$T(n)$のオーダーを求めよ。ただし$T(1)=1$とする。計算の過程も簡潔に記すこと。 (各 2 点)

1) $T(n)=4T(n/2)+n$
2) $T(n)=4T(n/2)+n^2$
3) $T(n)=3T(n/2)+n^2$
4) $T(n)=2T(n/2)+\log n$
5) $T(n)=2T(n/2)+n\log n$

問3　2-3 木について以下の質問に答えよ。(各 5 点)

1) 13, 31, 25, 33, 67, 16, 45 をこの順に挿入したときにできる２－３木を描け。
2) 上の２-３木から 33, 25 をこの順に削除したときにできる２－３木を描け。

問4　以下の擬似コードは有向グラフのある出発点からすべての頂点への最短経路を求めるアルゴリズムである。(※) に入るべき式を記せ。(10 点)

```
// G の頂点 s からの G の他の点 v への最短距離をあらわす D[v]を求める。
// C[a,b] は a から b への辺の長さ。辺がない場合は∞
dijkstra(頂点集合 V, 頂点間の距離行列 C, 出発点 s){
    S={s}
    for( v in V )
        D[v] = C[s, v];
    while( V-S ≠φ ){
        w は V-S の中で D[w]の値が最小となるものとする。
        S に w を付け加える。
        for( v in V-S )
            D[v] = (          ※          );
    }
    return D
}
```

問5　以下の擬似コードは 2 分探索木から最小の要素を削除し、その値を返すアルゴリズムである。(※) に入るべき処理の内容を簡潔に記せ。(10 点)

```
// 2 分探索木のノード node は、自身と同じ型のフィールド値として
// parent, leftchild, rightchild を持つとする。
deletemin(2 分探索木のノード node){
    if (node.leftchild == null){
            (          ※          )
            return node.value;
        }
    else
            return deletemin(a.leftchild);
}
```

問6　以下の擬似コードはソートをするアルゴリズムである。(※1)にはいるべき式、 (※2)に入るべき処理（1 命令とは限らない）の内容を簡潔に示せ。ただし、配列の長さを n とすると、配列の添え字は 0 から n-1 までとする。(各 5 点)

```
sort(配列 a){
    n = a の長さ;
    subsort(a, 0, n);
}

subsort(配列 a, int s, int e){
    for( pIndex = s+1; pIndex < e; pIndex = pIndex+1 ) {
        if( a[s] < a[pIndex] ) break;
        if( a[s] > a[pIndex] ){ pIndex = s; break; }
    }
    if( pIndex == e ) return;        // 全部が同じ値だった
    p = a[pIndex];
    i = s; j = e-1;
    while(true){
        while( a[i] < p ) i = i+1;
        while( a[j] >= p ) j = j-1;
        if(          ※1          ) break;
        swap(a[i], a[j]);       // a[i]と a[j]を交換する
    }
    (          ※2          )
}
```

問7　以下は整数を 10 を基数とする基数ソートの擬似コードである。(※1), (※2)に入るべき式を記せ。(各 5 点)

```
// Queue は FIFO であるデータ構造で、
// 順に値を入れる enqueue、順に値を取り出す dequeue が定義されている
sort(int [] a){
    Queue q[10];
    div = 1;
    n = a の長さ
    while(true){
        for(i = 0; i < n; i = i+1){
            index = (          ※1          )
            q[index].enqueue(a[i]);
        }
        if( q[0] の要素数 ==  n ) break;
        for(i = 0, j = 0; j < 10; j = j+1){
            while( q[j] is not empty ){
                a[i] = q[j].dequeue();
                i = i+1;
            }
        }
        div = (          ※2          )
    }
}
```

問8　double の優先度つき待ち行列をヒープを用いて実装したい。以下のインタフェースを持つ insert と deletemin を擬似コードであらわせ。ただし、配列 double [] heap と現在の要素数 int size は大域変数として与えられており、配列の長さは無限にあるとしてよい。また、deletemin の場合において heap が空であることは考えなくてもよい。配列の添え字は 0 から始まるものとする。(各 5 点)

- `void insert(double value);`　// 要素 value を待ち行列に追加する
- `double deletemin();`　// 現在の待ち行列の中で最小の値を削除し、その値を返す

問9　以下の擬似コードは、コスト最小の極大木を構成する辺の集合を求めるアルゴリズムである。(※1) (※2) に入るべき命令を記せ。(各 5 点)

```
// 辺は、端点 v1, v2 およびコスト w をもつ。
// 集合に要素を加える操作 add、
// 集合に要素があるかどうかを返す操作 contains がある
prim(頂点集合 V, 辺集合 E){
    T = {}; //辺の集合
    q = 辺のコストを重みとする優先度付待ち行列
    v = { V の何か要素１つ}
    S = {v} //頂点の集合
    while( S ≠ V ){
        for( edge in E ){
            if( edge.v1 == v || edge.v2 == v )
            q.insert(edge);
        }
        while(true){
            edge = q.deletemin(); // q の中でコストが最小のものを削除、取得する
            if ( not S.contains(edge.v1)) {
                    (       ※1       )
                    break;
            }
            else if ( not S.contains(edge.v2)) {
                    (       ※2       )
                    break;
            }
        }
        T.add(edge);
        S.add(v);
    }
    return T;
}
```

問10　以下の擬似コード divide は有向グラフの強連結成分分解を行うアルゴリズムであるが、誤りがある。誤りを訂正せよ。(10 点)
(ヒント　ある処理の行われる場所が間違っている)

// divide は、同じ強連結成分に含まれる頂点には同じ groupId をつける関数である
// 各頂点には、bool フィールド mark、整数フィールド number, groupId
// 各枝には頂点のフィールド、start、end を持ち start から end への有向枝を表す

```
整数 current = 0;

visit1(頂点 v, 枝集合 E){
    if( v.mark == true ) return;
    v.mark = true;
    v.number = current;
    current = current+1;
    for( e in E ){
        if( v != e.start ) continue;
        visit1(e.end, E);
    }
}

visit2(頂点 v, 枝集合 E){
    if( v.mark == true ) return;
    v.mark = true;
    v.groupId = current;
    for( e in E ){
        if( v != e.end ) continue;
        visit1(e.start, E);
    }
}
```

```
divide(頂点集合 V, 枝集合 E){
    for( v in V ){
        v.mark = false;
    }
    current = 0;
    for( v in V ){
        visit1(v, E);
    }
    for( v in V ){
        v.mark = false
    }
    V を、v.number の降順にソート
    current = 0;
    for( v in V ){
        if( v.mark == false ){
            visit2(v, E);
            current = current + 1;
        }
    }
}
```