This is the model solution for Homework F as well as grading policy.

Problem 9–2 part c.

Show how to compute the weighted median in Θ(n) worst–case time using a linear–time median algorithm such as Select from 9.3.

There were two common approaches to this problem: 1) writing an enclosing algorithm which used deterministic select to find the median of dataset and then recursively call this algorithm on the pertinent half of the dataset until the weighted median was identified, and 2) modifying deterministic select to include weights as in integral part of the first order selection process. Both are valid approaches, given the problem statement, and both have Θ(n) efficiency. The 2nd approach was the more difficult, but should result in a more efficient (by difference in coefficient) than the 1st approach.

As both approaches are valid, grading policy gave full credit for complete answers using either approach.

Grading policy gives:

1. Split was 6 points for the algorithm and 4 points for the corresponding analysis. Solutions without analysis lost 4 points in score.
2. (4) Lack of specificity on the algorithm lost up to 4 points.
3. (3) Not distinguishing < pivot from > pivot recursive calls lost 3 points.
4. (2) Not providing the correct argument for the cut–point lost 2 points.
5. (2) Not showing the calculation of sum of weights lost 2 points.
6. (1) Calculating weights over the whole range rather than just one side lost 1 point.
7. (*) No distinction was made between those who did or did not apply the modified algorithm to the medians by 5. This would affect average and worst case performance, and using the modified algorithm for the medians by 5 should give superior performance. However, this is too darned much to worry about in a single homework problem!
8. (3) Lack of specificity on the analysis lost up to 3 points.
9. (3) Lack recurrence equation lost 3 points.
10. (1) No reference to master theorem case (when used) lost 1 point.
11. (1) No consideration of initial condition (when appropriate) lost 1 point.
12. (3) Incorrect use of substitution or master theorem lost up to 3 points.

1st approach – enclosing algorithm

Algorithm:

```
Template<class Iterator>
Iterator selectWtMedian(Iterator b, Iterator e, double cutoff)
// b is beginning iteration on pairs
// e is just past final iteration on pairs
// cutoff if what percentile we are looking for.  Defaults to median at .5.
{
        size_t n = e–b;
        if ( n == 1) return b;   // one element is must be median
        Iterator q = deterministicSelect(b, e,  n/2);    // partitions dataset into 2 halves

        // select in proper segment of partition
        sum = 0.0;
        for (Iterator i = b; i < q; ++i)  sum += i.second;
        if ( sum > cutoff)
                // median is in lower partition
                return selectWtMedian( b, q, cutoff);
        else if (sum+q.second) < cutoff)
                // median is in upper partition
                // subtract wts already accounted for
                return selectWtMedian( q, e, cutoff – (sum+q.second));
        else    // it is at the partition!
                return q;
}
```

Analysis:

The deterministicSelect algorithm has been shown as $\Theta(n)$ in class and the text. Similary for $n < 140$ the efficiency of deterministicSelect is $\Theta(1)$.

For the above algorithm, the size of the dataset is reduced by half each time with additional summation of weights of $n/2$. The recurrence equation for the above algorithm is expressed as:

The claim for $f(n)$ takes into account the result from deterministicSelect and the additional work for summing weights above.

The master theorem, case 3, applies in this case with $a = 1$ and $b = 2$, and $\varepsilon=1$. This gives efficiency of selectWtMedian as $\Theta(n)$ for $n > 140$.

2nd Approach – modify deterministic select

This is a modification of "deterministicSelect" given as class handout Sep 25, 2003.

Algorithm–implementation:

```
Template<class Iterator>
Iterator deterministicSelectWt(Iterator b, Iterator e, double cutoff = 0.5)
// b is beginning iteration on pairs
// e is just past final iteration on pairs
// cutoff is the cumulative weight cutoff for this partition.  Default is 0.5 for median.
{
        size_t n = e–b;
        if ( n == 1) return b;   // one element and k must be 0
        // sort by groups of 5
        for (Iterator i = b; i < e; i +=5) insertionSort(i, min(i+5, e));

        // Find median of medians recursively
        // Use iterators that stride by 5
        FiveIter<Iterator> B(b+2);
        FiveIter<Iterator> E(e);
        // recognize that median is number at half of sum of wts.
        // this applies whether wts are 1 as for raw data (as in 9–2a) or
        // representative of distribution in population as in 9–2b.
        // Here wts are sample from dataset whose wts sum to 1.
        double sum = 0.0;
        for (Iterator i = b; i < E; ++i) sum +=i.second;
        FiveIter<Iterator> Q = deterministicSelectWt( B, E, sum/2);

        // partition on median of medians
        swap(*b, *Q);
        Iterator q = partition(b, e);

        // select in proper segment of partition
        sum = 0.0;
        for (Iterator i = b; i < q; ++i)  sum += i.second;
        if ( sum > cutoff)
                // median is in lower partition
                return deterministicSelectWt( b, q, cutoff);
        else if (sum+q.second) < cutoff)
                // median is in upper partition
                // subtract what wts are already accounted for
                return deterministicSelectWt( q, e, cutoff – (sum+q.second));
        else    // it must be at the partition!
                return q;
}
```

Analysis:

Since we have modified the deterministicSelect algorithm itself, our analysis has to start at that level of detail of the analysis done in class and in our text.

We modify our analysis from deterministicSelect to include the cost of summing weights of medians by 5 and the sum of weights from lower half of the partition:

The n/5 term takes into account the summation of weights for medians and the 7n/10 term includes the summation of weights in determining which partition to recurse on.

As neither of these terms is involved in the recurrence itself (i.e. the T() terms) and as they do NOT change the order of f(n), we conclude the efficiency of deterministicSelect is preserved as $\Theta(n)$.