MA U34605 Quiz 03 w/e 6/11/20 ANSWERS

Answer any 3 questions. Submit them using the `submit-work` program as pdfs, either handwritten and scanned, or typeset. They should be submitted before 1pm on Tuesday 17 November. There is an extra week because of Reading Week. All questions carry 20 marks.

(1) Show it is possible to sort n integers, all in the range $1 \dots n^7$, in $O(n)$ steps.

Answer. If $n \leq 1$ there is nothing to do. Otherwise convert each one of the integers x_i to $((x_i) - 1)_n$, a '7-digit' number with radix n. Sort these in $O(n)$ steps with radix sort and use the result to put the original numbers in sorted order.

(2) Show that every nonempty acyclic digraph G has a source.

Answer. Begin at any vertex u, and, while u is not a source, choose an edge (t, u) and replace u by t. If this procedure does not find a source then the graph is not acyclic.

(3) Show that if u is a source of G and G is *not* acyclic, then neither is the deleted graph $G \backslash u$.

Definition: given $G = (V, E)$, $u \in V$,

$$G \backslash u = (V \backslash u, \{(x, y) \in E : \ x \neq u \wedge y \neq u\}),$$

where $G = (V, E)$. It is obtained by deleting u and all edges incident to u.

More generally, given $X \subseteq V$, $G \backslash X$ is obtained by deleting all vertices in X and all edges incident to vertices in X.

Answer. No nontrivial directed cycle can contain u, since otherwise it would contain an edge into u and an edge out of u. So a nontrivial directed cycle in G is also a nontrivial directed cycle in $G \backslash u$.

(4) Let $G = (V, E)$ be a digraph. Define a relation $\sim$ on V by:

$u \sim v$ if and only if there exists a directed cycle containing both u and v; or, equivalently, there exist paths from u to v and from v to u.

Definition. A *path from* u to v in G is a sequence $u_1, \dots, u_k$ where $u_1 = u$, $u_k = v$, and, for $1 \leq j < k$, (u_j, u_{j+1}) is an edge of G.

Show that $\sim$ is an equivalence relation on V, i.e., reflexive, symmetric, and transitive.

Answer.

- Reflexive: $\{u\}$ is a path from u to itself.
- Symmetric: suppose $u \sim v$. There exists a nontrivial cycle containing both. Then $v \sim u$.
- Transitive: If there is a path from u to v and from v to w, then these have the form $u_1, \dots, u_k$ and $v_1, \dots, v_\ell$. If $k = 1$ then $u = v$ and there is a path from u to w.

Otherwise, $u_1, \ldots, u_{k-1}, v_1, \ldots, v_\ell$ is a path from u to w: if there is a path from u to v and v to w then there is a path from u to w.

Similarly, if there are paths from w to v and v to u then there is a path from w to u. This proves transitivity.

(5) MSD radix sort is more natural and more cumbersome than LSD. Show that, with k digits, its runtime is $O(kn)$. Here is a rough description of the algorithm.

First, it seems easiest to begin by copying the keys into a linked list, sorting the list, and copying back to an array. So we assume we are sorting a linked list.

```
To sort a list S of keys, sorting on k digits;
assume the radix is 10 for simplicity.

  Sort on the (k-1)-th digit, distributing
  the keys into 10 buckets (linked lists)
  according to their k-th digit. The 0-th
  digit is the low-order or 'units' digit.

  If k > 0, _recursively_ sort each bucket
  on k-1 digits.

  recombine the sorted buckets into a single
  list.
```

The question is to show that the cost of sorting n keys on k digits is $O(nk)$, assuming that the cost of distributing n keys into 10 buckets is $O(n)$ and the cost of recombining the buckets is $O(1)$ (there are only 10 buckets to be combined).

Answer. Induction. We may assume that the cost of sorting n keys on a single digit is $\leq an + b$, as is the total cost of distributing n keys and recombining the keys. Let $C(k, n)$ be the cost of sorting on k keys.

Given $k > 1$,

$$C(k, n) \leq an + b + \sum_{d=0}^{9} C(k-1, n_d)$$

where n_d is the number of keys whose $k-1$-st digit is d.

Appendix on lexical sorting.

```
My version of LSD sort has the following types and routines

typedef struct LIST_ELEMENT { int number; struct LIST_ELEMENT * next; }
  LIST_ELEMENT;
```

```
typedef struct { LIST_ELEMENT * first, * last; }
  BUCKET;

int digit ( int i, int radix, int number );
        Returns the i-th digit in the given radix
int length ( int radix, int number );
        Number of digits
void detach_list_element_to_bucket ( LIST_ELEMENT * le, BUCKET * bucket );
        Idea is that the array of numbers is first copied into
        a linked list.  'Detach' reminds one that le->next
        is set to NULL, and one needs to keep track of the
        annulled value.  This procedure adds a list element
        to a bucket.
void join_buckets ( BUCKET * tobucket, BUCKET * frombucket );
        Standard operation with linked lists.
void sort_bucket_on_digit ( int k, int radix, BUCKET * bucket );
        Creates an array buckets[radix], one for each digit
        (range 0 .. radix-1), transfers bucket elements to
        the appropriate buckets[d], and joins them together
        again, adjusting 'bucket' as necessary.

main program reads numbers into an array, copies them into
        a linked list, gets their maximum length maxlength,
        and sorts as follows:

  for (i=0; i<=maxlen; ++i)
    sort_bucket_on_digit ( i, 10, & bucket );
-----------------------------------

MSD radix sort uses the same list types and has the following
        routines
int digit ( int i, int radix, int number );
int length ( int radix, int number );
void detach_list_element_to_bucket ( LIST_ELEMENT * le, BUCKET * bucket );
void join_buckets ( BUCKET * tobucket, BUCKET * frombucket );
void sort_bucket_on_digit ( int k, int radix, BUCKET * bucket );
void msd_sort ( int k, int radix, BUCKET *bucket );

        The difference is that msd_sort is recursive and
sorts on the k low-order digits....

  if ( k > 0 )
  {
    for (i=0; i<radix; ++i)
      msd_sort ( k-1, radix, &buckets[i] );
```

```
  bucket->first = bucket->last = NULL;
  for (i=0; i<radix; ++i)
    join_buckets(bucket,&buckets[i]);
}
```