

## Irredundant Intervals

Donald E. Knuth

Computer Science Department, Stanford University

**Abstract.** This expository note presents simplifications of a theorem due to Győri and an algorithm due to Franzblau and Kleitman: Given a family  $F$  of  $m$  intervals on a linearly ordered set of  $n$  elements, we can construct in  $O(m + n)^2$  steps an irredundant subfamily having maximum cardinality, as well as a generating family having minimum cardinality. The algorithm is of special interest because it solves a problem analogous to finding a maximum independent set, but on a class of objects that is more general than a matroid. This note is also a complete, runnable computer program, which can be used for experiments in conjunction with the public-domain software of *The Stanford GraphBase*.

	Section	Page
Introduction .....	1	1
Theory .....	3	2
Practice .....	23	7
The main algorithm .....	26	9
The dénouement .....	32	12
Comments and extensions .....	38	15
References .....	44	16
Index .....	46	17

**1. Introduction.** Let's say that a family of sets is irredundant if its members can be arranged in a sequence with the following property: Each set contains a point that isn't in any of the preceding sets.

If  $F$  is a family of sets, we write  $F^\cup$  for the family of all nonempty unions of elements of  $F$ . When  $F$  and  $G$  are families with  $F \subseteq G^\cup$ , we say that  $G$  generates  $F$ . If  $F$  is irredundant and  $G$  generates  $F$ , we obviously have  $|F| \leq |G|$ , because each set in the sequence requires a new generator.

In the special case that the members of  $F$  are intervals of the real line, András Frank conjectured that the largest irredundant subfamily of  $F$  has the same cardinality as  $F$ 's smallest generating family. This conjecture was proved by Ervin Győri [4], who noted that such a result was a minimax theorem of a new type, apparently unrelated to any of the other well-known minimax theorems of graph theory and combinatorics. A constructive proof was found shortly afterwards by Franzblau and Kleitman [3], who sketched an algorithm to find a generating family and irredundant subfamily of equal cardinality. (Győri, Franzblau, and Kleitman were led to these results while studying the more general problem of finding a minimum number of subrectangles that cover a given polygon. Further information about polygon covers appears in [3] and [1].)

The purpose of this note is to describe the beautiful algorithm of Franzblau and Kleitman in full detail. Indeed, the CWEB source file that generated this document is a computer program that can be used in connection with the Stanford GraphBase [8] to find maximum irredundant subfamilies and minimum generating families of any given collection of intervals. Perhaps this new exposition will shed new light on the class of optimization problems for which an efficient algorithm exists.

According to the conventions of CWEB [9], the sections of this document are sequentially numbered 1, 2, 3, etc. In this respect we are returning to a style of exposition used by Euler and Gauss and their contemporaries. A CWEB program is also essentially a hypertext; therefore this document may also be regarded as experimental in another sense, as an attempt to find new forms of exposition appropriate to modern technology.

Note: Győri used the term "U-increasing" for an irredundant family; Franzblau and Kleitman called such intervals "independent." Since a family of sets is a hypergraph, it seems unwise to deviate from the standard meaning of independent edges, yet "U-increasing" is not an especially appealing alternative. We will see momentarily that the term "irredundant" is quite natural in theory and practice.

**2.** A far-reaching generalization of Győri's theorem was proved recently by Frank and Jordán [2], who introduced a large new family of minimax theorems related to linking systems. In particular, Frank and Jordán extended Győri's results to intervals on a circle instead of a line. But no combinatorial algorithm is known as yet for the circular case.

Can any or all of the Franzblau/Kleitman methods be "lifted" to such more general problems? We will return to this tantalizing question after becoming familiar with Franzblau and Kleitman's remarkable algorithm.

**3. Theory.** It is wise to study the theory underlying the Franzblau/Kleitman algorithm before getting into the program itself.

**4.** A family of sets is called *redundant* if it is not irredundant. Any family that contains a redundant subfamily is redundant, since any family contained in an irredundant family is irredundant.

**5.** If  $F$  is a family of sets and  $s$  is an arbitrary set, let  $F|s$  denote the sets of  $F$  that are contained in  $s$ . This operation is left-associative by convention:  $F|s|t = (F|s)|t = F|(s \cap t)$ .

We also write  $\bigcup F$  for  $\bigcup\{f \mid f \in F\}$ ; thus  $F|\bigcup F = F$ .

(An index to all the main notations and definitions that we will use appears at the end of this note.)

**6. Lemma.** A finite family  $F$  is redundant if and only if there is a nonempty set  $s$  such that every point of  $s$  belongs to at least two members of  $F|s$ . (The set  $s$  need not belong to  $F$ .)

Proof: If  $F$  is irredundant there is no such  $s$ , because  $F|s$  is irredundant and its last set in the assumed sequence contains a point that isn't in any of the others. But if  $F$  is redundant, it contains a minimal redundant subfamily  $F_0$ ; then we have

$$f \subseteq \bigcup (F_0 \setminus \{f\}) \quad \text{for all } f \in F_0,$$

since  $F_0 \setminus \{f\}$  is irredundant. It follows that every point of  $s = \bigcup F_0$  is contained in at least two members of  $F_0$ , hence in at least two members of  $F|s$  (since  $F_0 \subseteq F|s$ ).

**7. Corollary.** A finite family  $F$  of intervals on a line is redundant if and only if there is an interval  $s$  such that every point of  $s$  belongs to at least two intervals of  $F|s$ . (The set  $s$  need not belong to  $F$ .)

Proof: Intervals are nonempty. By the proof of the preceding lemma, it suffices to consider sets  $s$  that can be written  $\bigcup F_0$  for some minimal redundant subfamily  $F_0$ . In the special case of intervals,  $\bigcup F_0$  must be a single interval; otherwise  $F_0$  would not be minimal.

**8. Henceforth** we will restrict consideration to finite families  $F$  of intervals on a linearly ordered set. It suffices, in fact, to deal with integer elements; we will consider subintervals of the  $n$ -element set  $[0..n]$ . (The notation  $[a..b]$  stands here for the set of all integers  $x$  such that  $a \leq x < b$ .)

If  $F$  is a family of sets and  $x$  is a point, we will write  $N_x F$  for the number of sets that contain  $x$ . The corollary just proved can therefore be stated as follows: " $F$  is irredundant if and only if every interval  $s \subseteq \bigcup F$  contains a point  $x$  with  $N_x F|s \leq 1$ ." This characterization provides a polynomial-time test for irredundancy.

**9.** Irredundant intervals have an interesting connection to the familiar computer-science concept of *binary search trees* (see, for example, [7, §6.2.2]): A family of intervals is irredundant if and only if we can associate its intervals with a binary tree whose nodes are each labeled with an integer  $x$  and an interval containing  $x$ . All nodes in the left subtree of such a node correspond to intervals that are strictly less than  $x$ , in the sense that all elements of those intervals are  $< x$ ; all nodes in the right subtree correspond to intervals that are strictly greater than  $x$ . The root of the binary tree corresponds to the interval that is last in the assumed irredundant ordering. Its distinguished integer  $x$  is an element that appears in no other interval.

Given such a tree, we obtain a suitable irredundant ordering by traversing it recursively from the leaves to the root, in postorder [6, §2.3.1]. Conversely, given an irredundant ordering, we can construct a binary tree recursively, proceeding from the root to the leaves.

10. An example might be helpful at this point. Suppose  $n = 9$  and

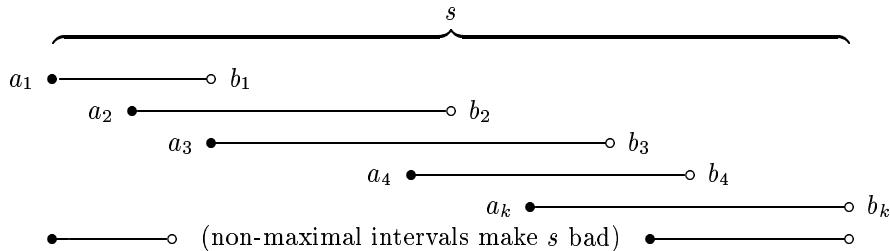
$$f_1 = [0..8], \quad f_2 = [0..7], \quad f_3 = [1..6], \quad f_4 = [1..5], \quad f_5 = [3..9], \quad f_6 = [2..9].$$

Then  $\{f_1, f_2, f_3, f_4, f_5\}$  and  $\{f_1, f_3, f_5, f_6\}$  are irredundant. (Indeed, a family of intervals is irredundant whenever its members have no repeated left endpoints or no repeated right endpoints.) These subfamilies are in fact maximally irredundant—they become redundant when any other interval of the family is added. Therefore maximal irredundant subfamilies need not have the same cardinality; irredundant subfamilies do not form the independent sets of a matroid.

On the other hand, irredundant sets of intervals do have matroid-like properties. For example, if  $F$  is irredundant and  $F \cup \{g\}$  is redundant, there is an  $f \neq g$  such that  $F \cup \{g\} \setminus \{f\}$  is irredundant. (The proof is by induction on  $|F|$ : There is  $x \in f \in F$  such that  $F = F_l \cup \{f\} \cup F_r$ , where  $F_l$  and  $F_r$  correspond to the left and right subtrees of the root in the binary tree representation. If  $x \in g$ , the family  $F_l \cup \{g\} \cup F_r$  is irredundant; if  $x < g$ , there is  $f' \in F_r$  with  $F_l \cup \{f\} \cup (F_r \cup \{g\} \setminus \{f'\})$  irredundant, by induction; if  $x > g$  there is  $f' \in F_l$  with  $(F_l \cup \{g\} \setminus \{f'\}) \cup \{f\} \cup F_r$  irredundant.) Such near-matroid behavior makes families of intervals especially instructive.

11. Let's say that an interval  $s$  is *good* for  $F$  if  $N_x F|s \leq 1$  for some  $x \in s$ ; otherwise  $s$  is *bad*. Franzblau and Kleitman introduced a basic reduction procedure for any family  $F$  of intervals that possesses a bad interval  $s$ . Their procedure is analogous to modification along an augmenting path in other combinatorial algorithms.

Let  $[a_1..b_1), \dots, [a_k..b_k)$  be the maximal intervals in  $F|s$ , ordered so that  $a_1 < \dots < a_k$  and  $b_1 < \dots < b_k$ . (Notice that  $s = [a_1..b_k)$ .) For example, we might have the following picture:



If  $a_{j+1} < b_j$  for  $1 \leq j < k$ , the family  $F$  reduced in  $s$  is defined to be

$$F \downarrow s = F \setminus \{[a_1..b_1), \dots, [a_k..b_k)\} \cup \{[a_2..b_1), \dots, [a_k..b_{k-1})\}.$$

In the simplest case we have  $k = 1$  and the reduced family is simply  $F \setminus \{[a_1..b_1)\}$ .

If  $s$  is a *minimal* bad interval for  $F$ , we can prove that  $a_{j+1} < b_j$  for  $1 \leq j < k$ , hence  $F \downarrow s$  is well-defined. Indeed, point  $a_{j+1}$  must be in some interval  $[c..d)$  other than  $[a_{j+1}..b_{j+1})$ , since  $s$  is bad. We can assume that  $c < a_{j+1}$ ; otherwise all intervals of  $F|s$  would be contained in  $[a_1..a_{j+1})$  or  $[a_{j+1}..a_k)$ , and both of these subintervals would be bad, contradicting the minimality of  $s$ . But  $c < a_{j+1}$  implies that  $[c..d)$  is contained in some maximal  $[a_i..b_i)$  with  $i \leq j$ . Hence  $a_{j+1} < b_i \leq b_j$ .

The notation  $F \downarrow s$  is defined to be left-associative, like  $F|s$ ; that is,  $F \downarrow s \downarrow t = (F \downarrow s) \downarrow t$  and  $F \downarrow s | t = (F \downarrow s) | t$ .

12. **Lemma.** *If  $s$  is a minimal bad interval for  $F$ , we have  $F \subseteq (F \downarrow s)^{\cup}$ .*

**Proof:** We must show that  $[a_j..b_j) \in (F \downarrow s)^{\cup}$  for  $1 \leq j \leq k$ . Let  $G = F \downarrow s | [a_j..b_j)$ ; we will prove that  $[a_j..b_j) = \bigcup G$ . If  $x \in [a_j..b_j)$ , the badness of  $s$  implies that  $x \in t$  for some  $t \in F|s$  with  $t \neq [a_j..b_j)$ ; let  $t$  be contained in the maximal interval  $[a_i..b_i)$ . If  $i = j$ , we have  $t \in G$ ; if  $i < j$ , we have  $x \in [a_j..b_i) \subseteq [a_j..b_{j-1}) \in G$ ; and if  $i > j$ , we have  $x \in [a_i..b_j) \subseteq [a_{j+1}..b_j) \in G$ .

**13. Lemma.** Suppose  $s$  is a minimal bad interval for  $F$ , while  $t$  is a good interval. Then  $t$  is good also for  $F \downarrow s$ .

Proof: Let  $s = [a..b]$ ,  $t = [c..d]$ ,  $l = \min(a, c)$ ,  $r = \max(b, d)$ . Suppose  $N_x F|[l..d] \geq 2$  and  $N_x F|[c..r] \geq 2$  for all  $x \in t$ . Then  $l = a < c < d < b = r$ , because  $t$  is good for  $F$ . By the minimality of  $s$ , there is some  $x \in [a..d]$  with  $N_x F|[a..d] \cdot x \leq 1$ . Since  $N_x F|[a..d] \geq 2$  we have  $x < c$ . Furthermore,  $x$  is in some interval of  $(F|s) \setminus F|[a..d]$ , because  $s$  is bad; so  $x$  is in some maximal  $[a_j..b_j]$  with  $a \leq a_j \leq x < c < d < b_j \leq b$ . It follows that none of the intervals  $[a_1..b_1], \dots, [a_k..b_k], [a_2..b_1], \dots, [a_k..b_{k-1}]$  are contained in  $t$ , hence  $F \downarrow s | t = F | t$ .

On the other hand, suppose  $N_x F|[l..d] \leq 1$  for some  $x \in t$ . We will show that  $N_x F \downarrow s | t \leq 1$ . This assertion can fail only if  $x$  lies in some interval  $[a_{j+1}..b_j] \subseteq t$  newly added to  $F \downarrow s$ . Then  $x \in [a_j..b_j] \subseteq [l..d]$ , and  $x \in [a_{j+1}..b_{j+1}] \notin [l..d]$ , hence  $b_j \leq d < b_{j+1}$ ; it follows that  $j$  is uniquely determined, and the only interval containing  $x$  in  $F \downarrow s | t$  is  $[a_{j+1}..b_j]$ . A similar argument applies if  $N_x F|[c..r] \leq 1$ .

**14. Corollary.** If  $s$  is a minimal bad interval for  $F$ , we have  $N_x F \downarrow s = N_x F - N_x s$ .

Proof: The proof of the preceding lemma shows in particular that none of the intervals  $[a_{j+1}..b_j]$  are already present in  $F$  before the reduction. And if  $x \in s$ , suppose  $x$  lies in  $[a_i..b_i], [a_{i+1}..b_{i+1}], \dots, [a_j..b_j]$ ; then it lies in  $[a_{i+1}..b_i], \dots, [a_j..b_{j-1}]$  after reduction, a net change of  $-1$ .

**15.** The Franzblau/Kleitman algorithm has a very simple outline: We let  $G_0 = F$  and repeatedly set  $G_{k+1} = G_k \downarrow s_k$ , where  $s_k$  is the leftmost minimal bad interval for  $G_k$ , until we finally reach a family  $G_r$  in which no bad intervals remain. This must happen sooner or later, because  $|G_k| = |F| - k$ . The final irredundant family  $G = G_r$  generates  $F$ , because  $F \subseteq G_k^{\cup}$  for all  $k$  by the lemma of §12. Franzblau and Kleitman proved the nontrivial fact that  $|G|$  is the size of the maximum irredundant subfamily of  $F$ ; hence  $G$  is a minimum generating family.

**16.** It is tempting to try to prove the optimality of  $G$  by a simpler, inductive approach in which we “grow”  $F$  one interval at a time, updating its maximum irredundant set and minimum generating set appropriately. But experiments show that the maximum irredundant set can change drastically when  $F$  receives a single new interval, so this direct primal-dual approach seems doomed to failure. The indirect approach is more difficult to prove, but no more difficult to program. So we will proceed to develop further properties of Franzblau and Kleitman’s reduction procedure [3]. The key fact is a remarkable theorem that we turn to next.

**17. Theorem.** The same final family  $G = G_r$  is obtained when  $s_k$  is chosen to be an arbitrary (not necessarily leftmost) minimal bad interval of  $G_k$  in the reduction algorithm. Moreover, the same multiset  $\{s_0, \dots, s_{r-1}\}$  of minimal bad intervals arises, in some order, regardless of the choices made at each step.

Proof: We use induction on  $r$ , the maximum number of steps to convergence among all reduction procedures that begin with a family  $F$ . If  $r = 0$ , the result is trivial, and if  $F$  has only one minimal bad interval the result is immediate by induction. Suppose therefore that  $s$  and  $t$  are distinct minimal bad intervals of  $F$ . We will prove later that  $t$  is a minimal bad interval for  $F \downarrow s$ , and that  $F \downarrow s \downarrow t = F \downarrow t \downarrow s$ . Let  $r'$  be the maximum distance to convergence from  $F \downarrow s$ , and  $r''$  the maximum from  $F \downarrow t$ ; then  $r'$  and  $r''$  are less than  $r$ , and induction proves that the final result from  $F \downarrow s$  is the final result from  $F \downarrow s \downarrow t = F \downarrow t \downarrow s$ , which is the final result from  $F \downarrow t$ . (Readers familiar with other reduction algorithms, like that of [5], will recognize this as a familiar “diamond lemma” argument. We construct a diamond-shaped diagram with four vertices:  $F$ ,  $F \downarrow s$ ,  $F \downarrow t$ , and a common outcome of  $F \downarrow s$  and  $F \downarrow t$ .) This completes the proof, except for two lemmas that will be demonstrated below; their proofs have been deferred so that we could motivate them first.

**18.** This theorem and the lemma of §13 have an important corollary: *Let  $S = \{s_0, \dots, s_{r-1}\}$  be the multiset of minimal bad intervals determined by the algorithm from  $F$ , and let  $t$  be any interval. Then  $S|t$  is the multiset of minimal bad intervals determined by the algorithm from  $F|t$ .* This holds because an interval  $s \subseteq t$  is bad for  $F$  if and only if it is bad for  $F|t$ . Minimal bad intervals within  $t$  never appear again once they are removed, and we can remove them first.

Reducing a minimal bad interval  $s$  when  $s$  is contained in a bad interval  $t$  may make  $t$  good, or leave it bad, or make it minimally bad. If  $s$  is minimally bad for  $F$ , it might also be minimally bad for  $F \downarrow s$ .

**19.** Now we are ready for the *coup de grâce* and the *pièce de résistance*: After the reduction algorithm has computed the irredundant generating family  $G = G_r$  and the multiset  $S$  of minimal bad intervals, we can construct an irredundant subfamily  $F'$  of  $F$  with  $|F'| = |G|$  by constructing a binary search tree as described in §9. The procedure is recursive, starting with an initial interval  $t = [0..n)$  that contains  $F$ : The tree defined for  $F|t$  is empty if  $F|t$  is empty. Otherwise it has a root node labeled with  $x$  and with any interval of  $F|t$  containing  $x$ , where  $x$  is an integer such that  $N_x G^{(t)} = 1$ ; here  $G^{(t)}$  is the final generating set that is obtained when the reduction procedure is applied to  $F|t$ . A suitable interval containing  $x$  exists, because every element of  $G^{(t)}$  is an intersection of intervals in  $F|t$ . The left subtree of the root node is the binary search tree for  $F|(t \cap [0..x))$ ; the right subtree is the binary search tree for  $F|(t \cap [x+1..n))$ .

The number of nodes in this tree is  $|G|$ . For if  $x$  is the integer in the label of the root,  $G$  has one interval containing  $x$ , and its other intervals are  $G|[0..x)$  and  $G|[x+1..n)$ . The family  $G^{(t)}$  is not the same as  $G|t$ ; but we do have  $|G^{(t)}| = |G|t|$  when  $t$  has the special form  $[0..x)$  or  $[x+1..n)$ , because in such cases  $F \downarrow s|t$  has the same cardinality as  $F|t$  when  $s$  is a minimal bad interval and  $s \not\subseteq t$ . For example, if  $t = [0..x)$  and  $b_j \leq x < b_{j+1}$ , we have  $F \downarrow s|t = F|t \setminus \{[a_1..b_1), \dots, [a_j..b_j)\} \cup \{[a_2..b_1), \dots, [a_{j+1}..b_j)\}$ .

**20.** It is not necessary to compute each  $G^{(t)}$  from scratch by starting with  $F|t$  and applying the reduction algorithm until it converges, because the binary tree construction algorithm requires only a knowledge of the incidence function  $N_x G^{(t)}$ . This function is easy to compute, because  $N_x F \downarrow s = N_x F - N_x s$  by §14; therefore

$$N_x G^{(t)} = N_x F|t - N_x S|t.$$

**21.** All the basic ideas of Franzblau and Kleitman's algorithm have now been explained. But we must still carry out a careful analysis of some fine points of reduction that were claimed in the proof of the main theorem. If  $s$  and  $t$  are distinct minimal bad intervals, the lemma of §13 implies that no bad subintervals of  $t$  appear in  $F \downarrow s$ ; we also need to verify that  $t$  itself remains bad.

**Lemma.** *If  $s$  is a minimal bad interval for  $F$  and  $t$  is a bad interval such that  $s \not\subseteq t$ , then  $t$  is bad for  $F \downarrow s$ .*

Proof: Let  $s = [a..b)$  and  $t = [c..d)$ . We can assume by left-right symmetry that  $a < c$ . Then  $b < d$ , by minimality of  $s$ . Assume that  $t$  isn't bad for  $F \downarrow s$ . The subfamily  $F|t$  must contain at least one of the maximal intervals  $[a_j..b_j)$  of  $F|s$  that are deleted during the reduction; hence  $c \leq a_j < b_{j-1} \leq b$ .

Let  $j$  be minimal with  $a_j \geq c$ . Then

$$F \downarrow s|t = F|t \setminus \{[a_j..b_j), \dots, [a_k..b_k)\} \cup \{[a_j..b_{j-1}), \dots, [a_k..b_{k-1})\};$$

so the elements of  $t$  that are covered once less often are the elements of  $[b_{j-1}..b_k)$ . Suppose  $y \in [b_{j-1}..b_i)$  for some  $i \geq j$ . Then  $y \in [a_i..b_i) \subseteq s \cap t$ , and  $y$  is in some other interval  $f \subseteq s$ . The maximal interval containing  $f$  must be  $[a_l..b_l)$  for some  $l \geq i$ , hence  $f \subseteq s \cap t$ . Thus  $N_y F|(s \cap t) \geq 2$  for all  $y \in [b_{j-1}..b_k)$ . But  $s \cap t$  is good, so there must be a point  $x \in [c..b_{j-1})$  with  $N_x F|(s \cap t) \leq 1$ . We also have  $N_x F|t \geq 2$ , since  $t$  is bad, so there's an interval in  $F|t \setminus F|(s \cap t)$  that contains  $x$ . This interval contains  $[x..b_k)$ . Consequently  $N_y F|t \geq 3$  for all  $y \in [b_{j-1}..b_k)$ .

**22. Lemma.** If  $s$  and  $t$  are minimal bad intervals for  $F$  and  $s \neq t$ , we have  $F \downarrow s \downarrow t = F \downarrow t \downarrow s$ .

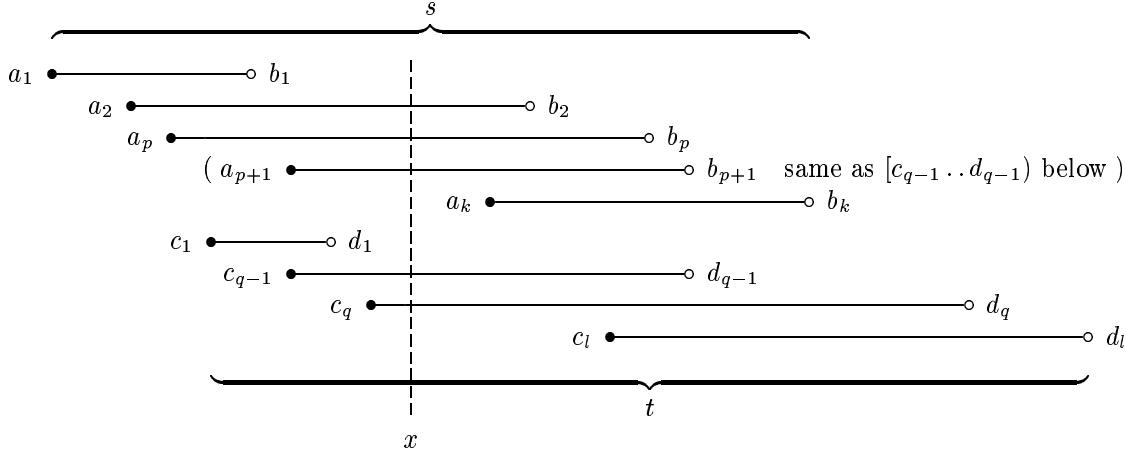
Proof: Let the maximal intervals in  $F|s$  and  $F|t$  be  $[a_1 \dots b_1], \dots, [a_k \dots b_k]$  and  $[c_1 \dots d_1], \dots, [c_l \dots d_l]$ , respectively, where  $a_1 < c_1$ . The lemma is obvious unless  $F|(s \cap t)$  is nonempty, so we assume that  $c_1 < b_k < d_l$ . Let  $x \in s \cap t$  have  $N_x F|(s \cap t) = 1$ , and let  $f$  be the interval of  $F|(s \cap t)$  that contains  $x$ . Let  $p$  be maximal with  $a_p < c_1$ , and let  $q$  be minimal with  $d_q > b_k$ . Since  $N_x F|s > 1$ , there is an interval  $[a_j \dots b_j]$  containing  $x$  with  $j \leq p$ ; thus  $x \in [a_p \dots b_p]$ . Similarly  $x \in [c_q \dots d_q]$ . Furthermore, if  $p < k$  we have  $[a_{p+1} \dots b_{p+1}] \subseteq s \cap t$ ; hence either  $[a_{p+1} \dots b_{p+1}] = f$  or  $a_{p+1} > x$ . If  $q > l$  we have either  $[c_{q-1} \dots d_{q-1}] = f$  or  $d_{q-1} \leq x$ .

If  $p = k$  or  $f \neq [a_{p+1} \dots b_{p+1}]$ , any newly added intervals  $[a_{j+1} \dots b_j]$  for  $p \leq j < k$  in  $F \downarrow s$  are properly contained in  $[c_q \dots d_q]$ , so they remain in  $F \downarrow s \downarrow t$ . Thus we can easily describe the compound operation  $F \downarrow s \downarrow t$  in detail:

Delete  $[a_1 \dots b_1], \dots, [a_k \dots b_k]$ ; insert  $[a_2 \dots b_1], \dots, [a_k \dots b_{k-1}]$ ;  
delete  $[c_1 \dots d_1], \dots, [c_l \dots d_l]$ ; insert  $[c_2 \dots d_1], \dots, [c_l \dots d_{l-1}]$ .

No two of these intervals are identical, so  $F \downarrow t \downarrow s$  gives the same result. (If  $f = [c_{q-1} \dots d_{q-1}]$ , the family  $F \downarrow t$  has  $f$  replaced by  $[c_q \dots d_{q-1}] \subseteq f \subseteq [a_i \dots b_i]$  for some  $i \leq p$ , so  $[c_q \dots d_{q-1}]$  is not maximal in  $F \downarrow t|s$ .)

The remaining case  $f = [a_{p+1} \dots b_{p+1}] = [c_{q-1} \dots d_{q-1}]$  needs to be considered specially, since we can't delete this interval twice. The following picture might help clarify the situation:



Suppose  $g$  is an interval of  $F|t$  that is contained in  $[c_j \dots d_j]$  if and only if  $j = q - 1$ . Then  $g$  contains a point  $< c_q$ . If  $x \in g$  we have  $g = f$ , since  $g \subseteq s \cap t$ . Otherwise  $g \subseteq [c_{q-1} \dots x] \subseteq [c_{q-1} \dots b_p] = [a_{p+1} \dots b_p]$ . It follows that the maximal intervals of  $F \downarrow s | t$  are

$[c_1 \dots d_1], \dots, [c_{q-2} \dots d_{q-2}], [c_{q-1} \dots b_p], [c_q \dots d_q], \dots, [c_l \dots d_l]$ .

These intervals are replaced in  $F \downarrow s \downarrow t$  by

$[c_2 \dots d_1], \dots, [c_{q-1} \dots d_{q-2}], [c_q \dots b_p], [c_{q+1} \dots d_q], \dots, [c_l \dots d_{l-1}]$ .

Thus  $F \downarrow s \downarrow t$  is formed almost as in the previous case, but with  $[a_{p+1} \dots b_p]$  and  $[c_{q-1} \dots d_{q-1}]$  replaced by  $[c_q \dots b_p]$ . And we get precisely the same intervals in  $F \downarrow t \downarrow s$ .

(Is there a simpler proof?)

**23. Practice.** The computer program in the remainder of this note operates on a family of intervals defined by a graph on  $n + 1$  vertices  $\{0, 1, \dots, n\}$ . We regard an edge between  $u$  and  $v$  as the half-open interval  $[u \dots v)$ , when  $u < v$ .

Graphs are represented as in the algorithms of the Stanford GraphBase [8], and the reader of this program is supposed to be familiar with the elementary conventions of that system.

The program reads two command-line parameters,  $m$  and  $n$ , and an optional third parameter representing a random-number seed. (The seed value is zero by default.) The Franzblau/Kleitman algorithm is then applied to the graph  $\text{random\_graph}(n + 1, m, 0, 0, 0, 0, 0, 0, 0, \text{seed})$ , a random graph with vertices  $\{0, 1, \dots, n\}$  and  $m$  edges. Alternatively, the user can specify an arbitrary graph as input by typing the single command-line parameter  $-g\langle \text{filename} \rangle$ ; in this case the named file should describe the graph in *save-graph* format (as in the MILES\_SPAN program of [8]).

When the computation is finished, a minimal generating family and a maximal irredundant subfamily will be printed on the standard output file.

If a negative value is given for  $n$ , the random graph is reversed from left to right; each interval  $[a \dots b)$  is essentially replaced by  $[-b \dots -a)$  (but minus signs are suppressed in the output). This feature lends credibility to the correctness of our highly asymmetric algorithm and program, because we can verify the fact that the minimum generating family of the mirror image of  $F$  is indeed the mirror image of  $F$ 's minimum generating family.

In practice, the algorithm tends to be interesting only when  $m$  and  $n$  are roughly equal. If  $n$  is large compared to  $m$ , we can remove any vertices of degree zero; such vertices aren't the endpoint of any interval. If  $m$  is large compared to  $n$ , we can almost always find  $n$  irredundant intervals by inspection. The running time in general is readily seen to be  $O(mn + n^2)$ .

```
#define panic(k)
{
    fprintf(stderr, "Oops, we're out of memory! (Case %d)\n", k);
    return k;
}

#include "gb_graph.h" /* the GraphBase data structures */
#include "gb_rand.h" /* the random_graph generator */
#include "gb_save.h" /* the restore_graph generator */

⟨Preprocessor definitions⟩

Graph *F; /* the graph that defines intervals */
Graph *G; /* a graph of intervals that generate F */
⟨Subroutines 33⟩

main(argc, argv)
    int argc; /* the number of command-line arguments, plus 1 */
    char *argv[]; /* the command-line arguments */
{
    register Vertex *t, *u, *v, *w, *x; /* current vertices of interest */
    register Arc *a, *b, *c; /* current arcs of interest */
    ⟨Scan the command-line options and generate F 24⟩;
    ⟨Compute G and S by the Franzblau/Kleitman algorithm 26⟩;
    if (gb_trouble_code) panic(1);
    ⟨Construct an irredundant subfamily of F with the cardinality of G 32⟩;
    ⟨Print the results 36⟩;
    return 0; /* this is the normal exit */
}
```

24. { Scan the command-line options and generate  $F$  24 }  $\equiv$

```

{
    int m = 0, n = 0, seed = 0;
    if (argc ≥ 3 ∧ sscanf(argv[1], "%d", &m) ≡ 1 ∧ sscanf(argv[2], "%d", &n) ≡ 1) {
        if (argc > 3) sscanf(argv[3], "%d", &seed);
        if (m < 0) {
            m = -m; /* we assume the user meant to negate n instead of m */
            n = -n;
        }
        if (n ≥ 0) F = random_graph(n + 1, m, 0, 0, 0, 0, 0, 0, 0, seed);
        else {
            G = random_graph(-n + 1, m, 0, 0, 0, 0, 0, 0, 0, seed);
            {Set F to the mirror image of G 25};
            gb_recycle(G);
        }
    }
    else if (argc ≡ 2 ∧ strncmp(argv[1], "-g", 2) ≡ 0) F = restore_graph(argv[1] + 2);
    else {
        fprintf(stderr, "Usage: %s [seed] | %s-gfoo.gb\n", argv[0], argv[0]);
        return 2;
    }
    if (!F) {
        fprintf(stderr, "Sorry, can't create the graph! (error code %ld)\n", panic_code);
        return 3;
    }
}
printf("Applying Franzblau/Kleitman to %s:\n", F->id);

```

This code is used in section 23.

25. { Set  $F$  to the mirror image of  $G$  25 }  $\equiv$

```

F = gb_new_graph(G->n);
if (!F) panic(4);
make_compound_id(F, "reflect(", G, ")");
for (v = G->vertices, u = F->vertices + F->n - 1; v < G->vertices + G->n; u--, v++) {
    v->clone = u;
    u->name = gb_save_string(v->name);
}
for (v = G->vertices; v < G->vertices + G->n; v++)
    for (a = v->arcs; a; a = a->next) gb_new_arc(v->clone, a->tip->clone, 1);

```

This code is used in section 24.

**26. The main algorithm.** We follow the outline of §15.

```
{ Compute  $G$  and  $S$  by the Franzblau/Kleitman algorithm 26 } ≡
  { Make  $G$  a copy of  $F$ , retaining only leftward arcs 27 };
  for ( $v = G\text{-vertices} + 2; v < G\text{-vertices} + G\text{-n}; v++$ )
    { Reduce all minimal bad intervals with right endpoint  $v$  and record them in  $S$  28 };
```

This code is used in section 23.

**27.** The algorithm doesn't need pointers from the left endpoint of an interval to the right endpoint; leftward pointers are sufficient. (This observation makes the reduction procedure faster.)

While copying  $F$ , we remove its rightward arcs, and we assign length 1 to all its leftward arcs. Later on, we will represent intervals of  $S$  by recording them in  $F$  as leftward arcs of length  $-1$ .

We also clear two utility fields of  $F$ 's vertices, since they will be used by the algorithm later. (They might be nonzero, if  $F$  was supplied with the  $-g$  option.)

```
#define clone  $u.V$  /* the vertex in  $F$  that matches a vertex in  $G$ , or vice versa */

{ Make  $G$  a copy of  $F$ , retaining only leftward arcs 27 } ≡
  switch_to_graph( $\Lambda$ ); /* prepare to return to graph  $F$  later */
   $G = gb\_new\_graph(F\text{-n});$  /* a graph with nameless vertices and no arcs */
  if ( $\neg G$ ) panic(5);
  for ( $u = F\text{-vertices}, v = G\text{-vertices}; u < F\text{-vertices} + F\text{-n}; u++, v++$ ) {
     $u\text{-clone} = v; v\text{-clone} = u;$ 
     $u\text{-y.I} = u\text{-z.I} = 0;$ 
     $v\text{-name} = gb\_save\_string(u\text{-name});$ 
    for ( $a = u\text{-arcs}, b = \Lambda; a; a = a\text{-next}$ )
      if ( $a\text{-tip} \geq u$ ) { /* we will remove the non-leftward arc  $a$  */
        if ( $b$ )  $b\text{-next} = a\text{-next};$ 
        else  $u\text{-arcs} = a\text{-next};$ 
      } else { /* we will copy the leftward arc  $a$  */
         $gb\_new\_arc(v, a\text{-tip}\text{-clone}, 0);$  /* the length in  $G$  is 0 */
         $a\text{-len} = 1;$  /* but in  $F$  the length is 1 */
         $b = a;$  /*  $b$  points to the last non-removed arc */
      }
    }
    if ( $gb\_trouble\_code$ ) panic(6);
    switch_to_graph( $F$ ); /* now we can add arcs to  $F$  again */
  }
```

This code is used in section 26.

**28.** Here's the most interesting part of the program, algorithmwise. Given a vertex  $v$ , we want to find the largest  $u$ , if any, such that  $[u \dots v)$  is bad for the intervals in  $G$ . So we sweep through the intervals  $[u \dots v)$  from right to left, decreasing  $u$  until it reaches a limiting value  $t$ . Here  $t$  is the least upper bound on a left endpoint that could guarantee double coverage of all points in  $[u \dots v)$ .

The utility field  $x\text{-}count$  records the number of intervals with left endpoint  $x$  and right endpoint  $w$  in the range  $u < w \leq v$ . Another utility field  $x\text{-}link$  is used to link vertices with nonzero counts together so that we can clear the counts to zero again afterwards.

It's easy to see that each iteration of the **while** loop in this section takes at most  $O(m + n)$  steps. The actual computation time is, however, usually much faster.

This program is designed to work correctly when  $G$  contains more than one arc from  $u$  to  $v$ . Duplicate arcs are discarded as a special case of the reduction procedure.

```
#define count z.I /* coverage decreases by this much when we pass to the left */
#define link y.V /* pointer to a vertex whose count field needs to be zeroed later */

{ Reduce all minimal bad intervals with right endpoint v and record them in S 28 }≡
{
    while (1) {
        int coverage = 0; /* the number of intervals ⊆ [t .. v) that contain u */
        int potential = 0; /* sum of x→count for x < t */
        Vertex *cleanup = Λ; /* head of the list of vertices with nonzero count */

        for (u = v, t = v - 1; u > t; u--) {
            coverage -= u→count; /* we prepare to decrease u */
            { Update the counts for all intervals ending at u 29 };
            if (coverage + potential < 2) { /* there's no bad interval ending at v */
                { Clean up all count fields 30 };
                goto done_with_v;
            }
        }
        while (coverage < 2) {
            t--;
            coverage += t→count;
            potential -= t→count;
        }
    }
    gb_new_arc(v→clone, u→clone, -1); /* [u .. v) is minimally bad; we record it in S = -F */
    { Replace G by G↓[u .. v) 31 };
    { Clean up all count fields 30 }; /* now we'll try again */
}
done_with_v: ;
}
```

This code is used in section 26.

**29.**  $\langle$  Update the counts for all intervals ending at  $u$  29  $\rangle \equiv$

```
for ( $a = u\rightarrow\text{arcs}$ ;  $a$ ;  $a = a\rightarrow\text{next}$ ) {
     $w = a\rightarrow\text{tip}$ ;
    if ( $w\rightarrow\text{count} \equiv 0$ ) {
         $w\rightarrow\text{link} = \text{cleanup}$ ;
         $\text{cleanup} = w$ ;
    }
     $w\rightarrow\text{count}++$ ;
    if ( $w \geq t$ )  $\text{coverage}++$ ;
    else  $\text{potential}++$ ;
}
```

This code is used in section 28.

**30.**  $\langle$  Clean up all  $\text{count}$  fields 30  $\rangle \equiv$

```
for ( $w = \text{cleanup}$ ;  $w$ ;  $w = w\rightarrow\text{link}$ )  $w\rightarrow\text{count} = 0$ ;
```

This code is used in section 28.

**31.** The reduction process is kind of cute too.

$\langle$  Replace  $G$  by  $G\downarrow[u..v)$  31  $\rangle \equiv$

```
for ( $a = v\rightarrow\text{arcs}$ ,  $c = \Lambda$ ,  $w = v$ ;  $a$ ;  $c = a$ ,  $a = a\rightarrow\text{next}$ )
    if ( $a\rightarrow\text{tip} \geq u \wedge a\rightarrow\text{tip} < w$ )  $w = a\rightarrow\text{tip}$ ,  $b = c$ ;
        /* now  $[w..v)$  is the longest interval from  $v$  inside  $[u..v)$ ; we'll remove it */
    if ( $b$ )  $b\rightarrow\text{next} = b\rightarrow\text{next}\rightarrow\text{next}$ ;
    else  $v\rightarrow\text{arcs} = v\rightarrow\text{arcs}\rightarrow\text{next}$ ;
        /* the remaining job is to shorten the other maximal arcs in  $[u..v)$  */
for ( $t = v - 1$ ;  $w > u$ ;  $t--$ ) {
    for ( $a = t\rightarrow\text{arcs}$ ,  $x = w$ ;  $a$ ;  $a = a\rightarrow\text{next}$ )
        if ( $a\rightarrow\text{tip} \geq u \wedge a\rightarrow\text{tip} < x$ )  $x = a\rightarrow\text{tip}$ ,  $b = a$ ;
            if ( $x < w$ )  $b\rightarrow\text{tip} = w$ ,  $w = x$ ; /*  $[x..t)$  is the longest interval from  $t$  */
}
```

This code is used in section 28.

**32. The dénouement.** Now we build a binary tree in the original graph  $F$ , by filling in some of the utility fields of  $F$ 's vertices. If a node in the tree is labeled with  $x$  and with the interval  $[u, v]$ , we represent it by  $x\rightarrow left = u$  and  $x\rightarrow right = v$ ; the subtrees of this node are  $x\rightarrow llink$  and  $x\rightarrow rlink$ . The root of the whole tree is  $F\rightarrow root$ .

The  $rlink$  field happens to be the same as the  $count$  field, but this is no problem because the  $rlink$  is never changed or examined until after the  $count$  has been reset to zero for the last time.

```
#define left x.V /* left endpoint of interval labeling this node */
#define right w.V /* right endpoint of interval labeling this node */
#define llink v.V /* left subtree of this node */
#define rlink z.V /* right subtree of this node */
#define root uu.V /* root node of the binary tree for this graph */

{ Construct an irredundant subfamily of  $F$  with the cardinality of  $G$  32 } ≡
  F→root = make_tree(F→vertices, F→vertices + F→n - 1);
```

This code is used in section 23.

**33.** With a little care we could maintain a stack within  $F$  itself, but it's easier to use recursion in C. Let's just hope the system programmers have given us a large enough runtime stack to work with.

This subroutine is based on the trick explained in §20.

```
{ Subroutines 33 } ≡
Vertex *make_tree(t, w)
  Vertex *t, *w;
{
  register Vertex *u, *v, *x;
  register Arc *a;

  { Find a vertex  $x$  with  $N_x F|[t..w] - N_x S|[t..w] = 1$  34 };
  if ( $\neg x$ ) return  $\Lambda$ ; /*  $F|[t..w]$  is empty */
  { Find an interval  $[u..v]$  such that  $x \in [u..v] \subseteq [t..w]$  35 };
  x→left = u;
  x→right = v;
  x→llink = make_tree(t, x);
  x→rlink = make_tree(x + 1, w);
  return x;
}
```

See also section 37.

This code is used in section 23.

**34.** A subtle bug is avoided here when we realize that a vertex might already be in the cleanup list when its *count* is zero.

```
(Find a vertex  $x$  with  $N_x F[t..w] - N_x S[t..w] = 1$ )  $\equiv$ 
{
    register int coverage = 0; /* coverage in  $F$  minus  $S$  */
    Vertex *cleanup = w + 1; /*  $w + 1$  is a sentinel value */
    for ( $v = w, x = \Lambda; v > t; v--$ ) {
        coverage -= v->count; /* now coverage refers to  $N_{v-1}$  */
        for ( $a = v->arcs; a; a = a->next$ ) {
            u = a->tip;
            if ( $u \geq t$ ) {
                if ( $u->link \equiv \Lambda$ )  $u->link = cleanup$ ,  $cleanup = u$ ;
                u->count += a->len; /* the length is +1 for  $F$ , -1 for  $S$  */
                coverage += a->len;
            }
        }
        if ( $coverage \equiv 1$ ) {
            x = v - 1; break;
        }
    }
    if ( $\neg x \wedge cleanup \leq w$ ) fprintf(stderr, "This can't happen!\n");
    while ( $cleanup \leq w$ ) {
        v = cleanup->link;
        cleanup->count = 0;
        cleanup->link =  $\Lambda$ ;
        cleanup = v;
    }
}
```

This code is used in section 33.

**35.**  $\langle$  Find an interval  $[u..v]$  such that  $x \in [u..v] \subseteq [t..w]$   $\rangle$   $\equiv$

```
for ( $v = w; v > x; v--$ ) {
    for ( $a = v->arcs; a; a = a->next$ ) {
        if ( $a->len > 0$ ) {
            u = a->tip;
            if ( $u \leq x \wedge u \geq t$ ) goto done;
        }
    }
done:
```

This code is used in section 33.

**36.**  $\langle$  Print the results  $\rangle$   $\equiv$

```
printf("Minimum_generating_family:");
for ( $v = G->vertices + 1; v < G->vertices + G->n; v++$ )
    for ( $a = v->arcs; a; a = a->next$ ) printf(" [%s .. %s]", a->tip->name, v->name);
printf("\nMaximum_irredundant_family:");
postorder_print( $F->root$ );
printf("\n");
```

This code is used in section 23.

37. There's just one subroutine to go. This is textbook stuff.

```
{ Subroutines 33 } +≡
void postorder_print(x)
    Vertex *x;
{
    if (x) {
        postorder_print(x→llink);
        postorder_print(x→rlink);
        printf(" □%s[%s..%s]", x→name, x→left→name, x→right→name);
    }
}
```

**38. Comments and extensions.** The program just presented incorporates several refinements to the implementation sketched by Franzblau and Kleitman in [3], and the author hopes that readers will enjoy finding them in the code. The Stanford GraphBase provides convenient data structures, by means of which it was possible to make the program short and sweet. However, most of the key ideas (except for the *make\_tree* procedure) can be found in [3].

**39.** Lubiw [10] discovered that the algorithm of Franzblau and Kleitman can be generalized so that it finds optimum irredundant subfamilies and generating families in an appropriate sense when the points of the underlying line have been given arbitrary nonnegative *weights*. It should be interesting and instructive to extend the program above so that it handles this more general problem.

**40.** The introductory remarks in §2 mention the recent breakthrough by Frank and Jordán [2], who showed (among many other things) that Győri's theorem can be extended to intervals on a circle as well as a line. Such a generalization was surprising because the size of a minimum generating set might be strictly larger than the size of a maximum irredundant subfamily of cyclic intervals. For example, the  $n$  intervals  $[k \dots k+2)$  for  $0 \leq k < n$  on the ring of integers mod  $n$  are obviously redundant; if we leave out any one of them, the remaining  $n - 1$  intervals will cover all  $n$  points. However, these cyclic intervals cannot be generated by fewer than  $n$  subintervals: No  $n - 1$  subintervals of length 1 will do the job, and if  $[k \dots k+2)$  is one of the generators the remaining  $n - 1$  intervals require  $n - 1$  further generators because they are irredundant.

Győri's minimax principle is restored, however, if we change the definition of irredundant families. We can say that  $F$  is irredundant if each  $f \in F$  has a distinguished element  $f_* \in f$ , such that whenever  $f$  and  $f'$  are distinct sets of  $F$  we have either  $f_* \notin f'$  or  $f'_* \notin f$ . If  $F$  is irredundant in this sense, and if  $G$  generates  $F$ , it is not difficult to prove that  $|F| \leq |G|$ : There is a  $g_f \in G$  for each  $f \in F$ , with the property that  $f_* \in g_f \subseteq f$ ; our new definition guarantees that  $g_f \neq g_{f'}$  when  $f \neq f'$ .

According to this new definition, the intervals  $[k \dots k+d)$  for  $0 \leq k < n$ , modulo  $n$ , are irredundant whenever  $n > 2(d-1)$ , because we can let  $[k \dots k+d)_* = k$ . Frank and Jordán showed that if  $F$  is any family of intervals modulo  $n$  with the property that each intersection  $f \cap f'$  of two of its members is either empty or a single interval, then the size of  $F$ 's smallest generating family is the size of its largest irredundant subfamily under the new definition.

**41.** For intervals on a line, Győri [4] had already observed that both definitions of irredundancy are equivalent. Suppose a system of representatives  $f_* \in f$  is given for all  $f$  in some family  $F$  of intervals on a line, such that  $f \neq f'$  implies  $f_* \notin f'$  or  $f'_* \notin f$ . If we cannot arrange those intervals in a sequence  $f^{(1)}, f^{(2)}, \dots, f^{(n)}$  such that  $f_*^{(j)} \notin f^{(1)} \cup \dots \cup f^{(j-1)}$  for  $1 < j \leq n$ , there must be some cycle of intervals such that  $f_*^{(1)} \in f^{(2)}, f_*^{(2)} \in f^{(3)}, \dots, f_*^{(m)} \in f^{(1)}$ , where  $m > 2$ . Consider the shortest such cycle, and suppose  $f_*^{(1)} = \min_{j=1}^m f_*^{(j)}$ . We cannot have  $f_*^{(k-1)} < f_*^{(k)}$  for  $1 < k \leq m$ , because  $f_*^{(m)} \in f^{(1)}$ ; let  $k > 1$  be minimum such that  $f_*^{(k-1)}$  is not strictly less than  $f_*^{(k)}$ . Then  $f_*^{(k-1)}$  must be strictly greater than  $f_*^{(k)}$ , and we have  $f_*^{(1)} < f_*^{(k)} < f_*^{(k-1)}$ . There is some  $j$  with  $1 < j < k$  and  $f_*^{(j-1)} < f_*^{(k)} < f_*^{(j)}$ ; since  $f_*^{(j-1)} \in f^{(j)}$  we have  $f_*^{(k)} \in f^{(j)}$ , a shorter cycle. This contradiction shows that no cycles exist.

**42.** Frank and Jordán gave another criterion for irredundancy that works also for general families of intervals on a circle when large intervals might wrap around so that their intersection  $f \cap f'$  consists of two disjoint intervals. In such cases they allow  $\{f_*, f'_*\} \subseteq f \cap f'$ , but only if  $f_*$  and  $f'_*$  lie in different components of  $f \cap f'$ . For example, the intervals  $[k \dots k+d)$  for  $0 \leq k < n$ , modulo  $n$ , are irredundant by this definition for all  $n > d$ . Once again the minimax theorem for generating families and irredundant subfamilies remains valid, in this extended sense.

**43.** The algorithms presented by Frank and Jordán [2] for such problems require linear programming as a subroutine. Therefore it would be extremely interesting to find a purely combinatorial procedure, analogous to the algorithm of Franzblau and Kleitman, either for the wrap-restricted situation of §40 or for the more general setup of §42.

**44. References.**

- [1] Joseph C. Culberson and Robert A. Reckhow, “Covering polygons is hard,” *Journal of Algorithms* **17** (1994), 2–44.
- [2] András Frank and Tibor Jordán, “Minimal edge-coverings of pairs of sets,” *Journal of Combinatorial Theory* **B65** (1995), 73–110.
- [3] Deborah S. Franzblau and Daniel J. Kleitman, “An algorithm for constructing polygons with rectangles,” *Information and Control* **63** (1984), 164–189.
- [4] Ervin Győri, “A minimax theorem on intervals,” *Journal of Combinatorial Theory* **B37** (1984), 1–9.
- [5] Donald E. Knuth and P. B. Bendix, “Simple word problems in universal algebras,” in *Computational Problems in Abstract Algebra*, edited by J. Leech (Oxford: Pergamon, 1970), 263–297. Reprinted in *Automation of Reasoning*, edited by Jörg H. Siekmann and Graham Wrightson, **2** (Springer, 1983), 342–376.
- [6] Donald E. Knuth, *Fundamental Algorithms*, Volume 1 of *The Art of Computer Programming* (Reading, Massachusetts: Addison-Wesley, 1968), xxii + 634 pp.
- [7] Donald E. Knuth, *Sorting and Searching*, Volume 3 of *The Art of Computer Programming* (Reading, Massachusetts: Addison-Wesley, 1973), xii + 722 pp. + foldout illustration.
- [8] Donald E. Knuth, *The Stanford GraphBase*: A Platform for Combinatorial Computing (New York: ACM Press, 1993), viii+576 pp. Available via anonymous ftp from `labrea.stanford.edu` in directory `~ftp/pub/sgb`.
- [9] Donald E. Knuth and Silvio Levy, *The CWEB System of Structured Documentation* (Reading, Massachusetts: Addison-Wesley, 1994).
- [10] Anna Lubiw, “A weighted min-max relation for intervals,” *Journal of Combinatorial Theory* **B53** (1991), 173–194.

**45.** The author thanks the referees of this note for many valuable suggestions that greatly improved the presentation.

#### 46. Index.

- a:* 23, 33.  
*abnormal exit 1:* 23.  
*abnormal exit 2:* 24.  
*abnormal exit 3:* 24.  
*abnormal exit 4:* 25.  
*abnormal exit 5:* 27.  
*abnormal exit 6:* 27.  
*arcs:* 25, 27, 29, 31, 34, 35, 36.  
*argc:* 23, 24.  
*argv:* 23, 24.  
*b:* 23.  
*bad interval, definition:* 11.  
*Bendix, Peter Bernard:* 44.  
*binary search tree, definition:* 9.  
*c:* 23.  
*cleanup:* 28, 29, 30, 34.  
*clone:* 25, 27, 28.  
*count:* 28, 29, 30, 32, 34.  
*coverage:* 28, 29, 34.  
*Culberson, Joseph Carl:* 44.  
*diamond lemma:* 17.  
*done:* 35.  
*done\_with\_v:* 28.  
*F:* 23.  
*fprintf:* 23, 24, 34.  
*Frank, András:* 1, 2, 40, 42, 43, 44.  
*Franzblau, Deborah Sharon:* 1, 11, 15, 16, 38, 43, 44.  
*G:* 23.  
*gb\_new\_arc:* 25, 27, 28.  
*gb\_new\_graph:* 25, 27.  
*gb\_recycle:* 24.  
*gb\_save\_string:* 25, 27.  
*gb\_trouble\_code:* 23, 27.  
*generating family, definition:* 1.  
*good interval, definition:* 11.  
*Györi, Ervin:* 1, 44.  
*id:* 24.  
*irredundant sets, definition:* 1, 40, 42.  
*Jordán, Tibor:* 2, 40, 42, 43, 44.  
*Kleitman, Daniel J [Isaiah Solomon]:* 1, 11, 15, 16, 38, 43, 44.  
*Knuth, Donald Ervin:* 38, 44, 45.  
*left:* 32, 33, 37.  
*len:* 27, 34, 35.  
*Levy, Silvio Vieira Ferreira:* 44.  
*link:* 28, 29, 30, 34.  
*llink:* 32, 33, 37.  
*Lubiw, Anna:* 44.  
*m:* 24.  
*main:* 23.
- make\_compound\_id:* 25.  
*make\_tree:* 32, 33, 38.  
*matroids:* 10.  
*metaphors, mixed:* 19.  
*n:* 24.  
*name:* 25, 27, 36, 37.  
*next:* 25, 27, 29, 31, 34, 35, 36.  
*notation: [a..b]:* 8.  
*notation:  $\bigcup F$ :* 5.  
*notation:  $F|s$ :* 5.  
*notation:  $F \downarrow s$ :* 11.  
*notation:  $F^\cup$ :* 1.  
*notation:  $G^{(t)}$ :* 19.  
*notation:  $N_x F$ :* 8.  
*Oops, we're out of memory:* 23.  
*panic:* 23, 25, 27.  
*panic\_code:* 24.  
*postorder\_print:* 36, 37.  
*potential:* 28, 29.  
*printf:* 24, 36, 37.  
*random\_graph:* 23, 24.  
*Reckhow, Robert Allen:* 44.  
*redundant sets, definition:* 4.  
*restore\_graph:* 23, 24.  
*right:* 32, 33, 37.  
*rlink:* 32, 33, 37.  
*root:* 32, 36.  
*save\_graph:* 23.  
*seed:* 23, 24.  
*Sorry, can't create the graph:* 24.  
*sscanf:* 24.  
*Stanford GraphBase:* 1, 23, 38.  
*stderr:* 23, 24, 34.  
*strcmp:* 24.  
*switch\_to\_graph:* 27.  
*t:* 23, 33.  
*This can't happen:* 34.  
*tip:* 25, 27, 29, 31, 34, 35, 36.  
*u:* 23, 33.  
*Usage:* 23, 24.  
*uu:* 32.  
*v:* 23, 33.  
*vertices:* 25, 26, 27, 32, 36.  
*w:* 23, 33.  
*x:* 23, 33, 37.

⟨ Clean up all *count* fields 30 ⟩ Used in section 28.  
⟨ Compute  $G$  and  $S$  by the Franzblau/Kleitman algorithm 26 ⟩ Used in section 23.  
⟨ Construct an irredundant subfamily of  $F$  with the cardinality of  $G$  32 ⟩ Used in section 23.  
⟨ Find a vertex  $x$  with  $N_x F|[t..w] - N_x S|[t..w] = 1$  34 ⟩ Used in section 33.  
⟨ Find an interval  $[u..v]$  such that  $x \in [u..v] \subseteq [t..w]$  35 ⟩ Used in section 33.  
⟨ Make  $G$  a copy of  $F$ , retaining only leftward arcs 27 ⟩ Used in section 26.  
⟨ Print the results 36 ⟩ Used in section 23.  
⟨ Reduce all minimal bad intervals with right endpoint  $v$  and record them in  $S$  28 ⟩ Used in section 26.  
⟨ Replace  $G$  by  $G \downarrow [u..v]$  31 ⟩ Used in section 28.  
⟨ Scan the command-line options and generate  $F$  24 ⟩ Used in section 23.  
⟨ Set  $F$  to the mirror image of  $G$  25 ⟩ Used in section 24.  
⟨ Subroutines 33, 37 ⟩ Used in section 23.  
⟨ Update the counts for all intervals ending at  $u$  29 ⟩ Used in section 28.

## Đề tài 18 Irredundant Intervals

(Tạm dịch: Những khoảng tối thiểu).

### \* Nội dung của bài bài

Bài bài minh họa việc đơn giản hóa của một đồ thị  
do Götz và một thuật toán của Franzblau và Kleitman.  
cho một họ  $A$  gồm các khoảng  $m$  trên một tập hợp  $n$   
phản xỉ được sắp xếp nguyên tố, chúng ta có thể  
xây dựng trong  $O(m+n)^2$  bước một phản họ có số  
phản xỉ tối đa, cũng như một tao ra một họ với  
số phản xỉ tối thiểu. Thuật toán chiết quan tâm đến bước  
vì nó giải quyết vấn đề như việc tìm kiếm một tập hợp  
đã lặp tối đa.

Bà sủ minh họa các tập hợp là tối thiểu nếu các thành  
phần của nó có thể được sắp xếp theo một dây với các thuộc tính  
sau: Mọi tập hợp chỉ một điểm không có trong bao bì tập hợp  
nào trước đó.

Một thuật toán của Franzblau và Kleitman mới  
còn chưa rõ, EWEB đã tạo ra, mục đích của bài tài liệu này  
là ghi lại về chương trình để tìm các phản họ là tối thiểu  
tối đa và các họ là tối thiểu của bài by tập hợp khoảng  
nhất định nào.

- redundant : (mathematics) . Containing no redundant constraint: không dư thừa, tối thiểu

- arranged: to plan, prepare for, or organize something: sắp xếp

- sequence: a series of related things or events, on the order in which they follow each other: chuỗi

- preceding: existing or happening before someone or something else

- generate: to cause something to exist: tạo ra

- subfamily: phân họ

- conjecture: a guess about something base on how it seems and not on proof: phỏng đoán

- cardinality: the number of elements (= separate items) in a mathematical set

- apparently: used to say you have read or been told something although you are not certain it is true: rõ ràng

- sketched: a short written or spoken story that does not have many details: phác thảo

- algorithm: a set of mathematical instruction or rules that, especially if given to a computer, will help to calculate an answer to a problem: thủ thuật toán

- sub rectangles: hình chữ nhật bên trong

- polygon: a flat shape with three or more straight sides: <sup>đường</sup> giác

tối ưu hóa

- optimization: the act of making something as good as possible.
- efficient: working or operating quickly and effectively in an organized way: có hiệu quả
- contemporary: existing or happening now, and therefore seeming modern: thời đại
- hypertext: a way of joining a word or image to another page, document, etc. on the internet or in another computer program so that you can move from one to the other easily. sự kết nối
- deviate: to do something that is different from the usual or common way of behaving: đi chệch hướng
- edge: the outer or furthest point of something: bờ
- momentary: lasting for a very short time: tạm thời
- far-reaching: something far-reaching has a great influence on many people or things: sâu rộng
- generalization: a written or spoken statement in which you say or write that something is true all of the time when it is only true some of the time: suy luận片面
- tantalizing: something that is tantalizing causes desire and excitement in you, but is unlikely to provide a way of satisfying that desire: trêu ngứa
- remarkable: unusual or special and therefore surprising and worth mentioning: đáng chú ý

- theory: a formal statement of the rules on which a subject of study is based or ideas that are suggested to explain a fact or event or, more generally, an opinion or explanation **học thuyết**
- redundant: (especially of a word, phrase, etc.) unnecessary because it is more than is needed. **thừa**
- arbitrary: base on chance rather than being planned or based on reason. **bất kỳ**
- denote: to represent something. **chỉ định**
- left-association: **liên kết bên trái**
- convention: a large formal meeting of people who do a particular job or have similar interest, or a large meeting for a political party. **giúp việc**
- lemma: a form of a word that appears as an entry in dictionary and is used to represent all the other possible forms. **bản đệm**
- finite: having a limited or end. **còn hạn**
- assume: to accept something to be true without question or proof. **thử nhận**
- Hence: that is the reason or explanation for. **vì thế**
- corollary: something that result from something else. **kiết quả**

- suffices: to be enough - đủ
- henceforth: starting from this time: từ nay đi
- restrict to limit the movements or actions of someone, or to limit something and reduce its size or prevent it from increasing - hạn chế, giới hạn, thu hẹp
- linearly: in a straight or nearly straight line: tuyến tính
- subinterval: any of several smaller intervals into which a larger one is divided: phân đoạn
- node: a place where things such as lines or systems join, nút
- strictly: in a very limited or limiting way: nghiêm ngặt
- matroid-like: a matroid is a finite set together with a generalization of a concept from linear algebra that satisfies a natural set of properties for that concept - một matroid là một tập hợp hữu hạn cùng với một khái niệm tổng quát của một khái niệm mà đây sẽ tuân thủ và duy trì nó
- nhanh - as though this is what the person intended
- correspond (to), to match or be similar or equal - khớp
- possesses to have or own something, or to have a particular quality - sở hữu
- procedure: an order or method of doing something - thủ tục

- analogous: having similar features to another thing and therefore able to be compared with it - số nh ng th.
- mortification: a change to something, usually to improve it - s đ nh đ nh th.
- augment: to increase the size or value of something by adding something to it - t nh l
- assertion: to say that something is certainly true. s h nh đ nh
- nontrivial nontrivial: having some variables or terms that are not equal to zero or an identity - c o m đ nh kh ng b nh O ho ac nh.
- temptation: to make someone want to have or do something, especially something that is more unnecessary or wrong - c u đ nh
- optimality: most favourable or desirable - s đ nh th
- inductive: using a particular set of facts or ideas to form a general principle - q uy n
- drastically: in a way that is severe and sudden or has very noticeable effects - s đ nh nh ng nh nh
- arbitrary: not decided by rules or laws but by a person's own opinion - ch u ng nh, nh nh đ nh.

- convergence: the fact that two or more things, ideas, etc. become similar or come together. *sự hội tụ*
- diagram: a simple plan that represents a machine, system, or idea, etc., often drawn to explain how it works. *bản đồ*
- defer: to put off to another time - *đi hoãn*.
- coup de grâce: an action that ends something that has been gradually getting worse, or that kill a person or animal in order to end their suffering - *tấn án*
- pièce de résistance - the best and most important or exciting thing, often the last in a series of things. *tái phán*, *sự kiện*
- recursive: involving doing or saying the same thing several several times over to produce a particular result or of on effect. *đóng lặp*.
- Compound: composed of a number of parts - *phối hợp*

### 3. Dịch nghĩa từng câu:

#### I. Giới thiệu

- Câu 1: Một họ các tập hợp được cho là tối thiểu nếu các thành phần của nó có thể được xếp thành một chuỗi với thuộc tính sau: Mọi tập hợp chứa một điểm mà điểm đó không có trong ~~các~~<sup>bất kỳ</sup> tập hợp nào trước đó.
- Câu 2: Nếu  $F$  là một họ của các tập hợp ta viết  $F^U$  là hiệu cho họ của tất cả các tập hợp khai rộng của  $F$ .
- Câu 3: Khi  $F$  và  $G$  là các họ với  $F \subseteq G^U$ , ta nói là  $G$  tạo ra  $F$ . Nếu  $F$  là tối thiểu và  $G$  tạo ra  $F$ , ta thấy rõ ràng rằng  $|F| \leq |G|$ , bởi vì mỗi tập trong chuỗi cần có một tập sinh.
- Câu 5: Trong trường hợp đặc biệt thì các thành phần của  $F$  là ~~các~~<sup>tổng</sup> khảng của đường thẳng Euclidean, András Frank phong đoán rằng ~~tập~~ phần họ tối thiểu lớn nhất có cùng bùn chửi với họ được tạo ra nhỏ nhất của  $F$ .
- Câu 6: Phong đoán này đã được chứng minh bởi Ervin Győri, ông đã chứng minh kết quả như vậy là một định lý minimax về một khai mở, đường như không liên quan đến bất kỳ định lý minimax nào tiếng nòi khác của lý thuyết đồ thị và tổ hợp.

- Câu 7. Một bằng chứng mang tính xác định đã được tìm thấy ngày sau đó bởi Franzblau và Kleinman, nhưng người ta phác thảo một thuật toán để tìm một họ sinh và phân họ tối thiểu có số lượng bùn chát bằng nhau
- Câu 8. Györi, Franzblau và Kleinman đã đưa ra đến những kết quả này trong khi nghiên cứu ~~ở~~ vấn đề tổng quát hơn ~~ở~~ là tìm ra số lượng tối thiểu các him chui nhất con, bao gồm phủ một địa giới nhất định
- Câu 9 Thông tin thêm về sự bảo phủ của giác xuất hiện trong [3] và [1]
- Câu 10 Mục đích của ghi chú này là mô tả thuật toán tiếp với của Franzblau và Kleinman một cách chi tiết
- Câu 11 Thật vậy, tạp chí CNET đã tạo tài liệu này là một chương trình máy tính có thể sử dụng liên quan đến Standford GraphBase [7] để tìm luồng tối đa xác minh họ tối thiểu <sup>lồng tối thiểu</sup> và xác họ sinh nhỏ nhất của bất kỳ tập hợp khoảng nhất định nào.
- Câu 12 Có thể sử dụng bài toán này để làm sang trọng và dễ dàng tối ưu hóa mà một thuật toán hiệu quả trên tai.

+) Câu 13: Theo quy tắc của CWEB [9], các phần của tài liệu này sẽ được đánh số tuân túc 1, 2, 3, v.v.

- Câu 14: Lỗi này, chúng ta đang quay tròn bởi một phong cách trình bày khác, đang bởi Euler & Gauss và những người cùng thời với họ

- Câu 15: Một chương trình CWEB có cơ bản cũng là tên văn bản, Do đó từ biến này cũng có thể được coi là thử nghiệm thao mờ nhòe khác, như mô hình để tìm ra các hình thái giải bài mới phù hợp với công nghệ hiện đại.

- Câu 16: Lỗi k: Gửi đã sử dụng một thuật ngữ " $U$ " <sup>increasing</sup> cho một họ  $U$  (hieu), Pranzblina. Khi mà nó không bằng như vậy là tôi gặp

- Câu 17: V) một họ các tập hợp là một siêu tập thì có vẻ như không bayati chính Kofly nghĩa là chuẩn của các cách abc lặp, nhưng " $U$ " <sup>increasing</sup> không phải là một xu hướng đặc biệt hấp dẫn

- Câu 18: Chú ý ta sẽ thấy trong giấy rất ràng thuật ngữ " $\text{increasing}$ " (không đủ thừa tối thiểu) là khá tự nhiên trong lý thuyết và thực hành

## II. Một số câu hỏi

- Câu 1: Một khái quát của công thức định lý [Gödel] đã được chứng minh gần đây bởi Frank và Jordan [P1], những người đã giải thích một cách định lý với tên minimax như là lén lén qua từ sau và thông tin bài.
- Câu 2: Đặc biệt, Frank và Jordan đã mô tả kết quả của lý thuyết thành các khung trắc nghiệm song tròn thay vì một đường thẳng.
- Câu 3: Những khung có thuật toán +S hợp nào chưa biết đến cho trường hợp tròn.
- Câu 4: Bài học hoặc tất cả các phương pháp Fronzblau, Kleiman có thể được nâng lên thành các vấn đề chung hơn như vậy không?
- Câu 5: Chúng ta có thể bài câu hỏi trên ngược lại sao khi làm quen với thuật toán tam giác của Fronzblau và Kleiman.

## III. Lý thuyết

- Câu 1: Thủ thuật ngoài khía cạnh cứu lý thuyết có bài câu hỏi Fronzblau và Kleiman-Froelich đi với nhau trình.

## IV

Câu 1: Một bài của cái tập hợp gọi là dãy thừa nếu nó không phải là số nguyên.

Câu 2: Bất cứ bộ nào nếu có chứa một phần bộ dãy thừa đều là dãy thừa, vì bất kỳ bộ nào có không một bộ tử thõa điều đều không là dãy thừa.

#### V. Mô hình F

- Câu 1: Nếu  $F$  là bộ của các tập hợp và  $S$  là một tập tùy ý, Ký hiệu  $F(S)$  biểu thị các tập hợp  $F$  dãy chia thành  $S$ .

- Câu 2: Tập  $T$  là mảng  $\{t_1, t_2, \dots, t_n\}$  với  $t_i \in S$  ( $i = 1, 2, \dots, n$ )

- Câu 3: Chứng ta có thể viết  $UF$  cho  $U(S) \subseteq F(S)$ ,  
thus  $F(UF) = F$

- Câu 4: Mô chí mục cho tất cả các Ký hiệu chính có xu hướng định nghĩa mà chúng ta sẽ sử dụng sau này ở cuối ghi chú này.

#### VI. Bộ đề

- Câu 1: Một bộ hữu hạn  $F$  là dãy thừa nếu và chỉ nếu có một tập hợp không có giá trị nào cho mỗi phần tử thuộc  $S$  thuộc ít nhất 2 thành phần của  $F$ .

- Câu 2: Tập hợp  $S$  không cần huỷ bỏ  $F$ .

- Câu 3: Chứng minh  $N_{\epsilon}(F)$  là không thuât thi  
Không có  $s$  như vậy, bởi vì  $F$  là tập  $\epsilon$ -thiểu và tập  
hợp  $s$  với  $c$  là  $\epsilon$ -thiểu. Trong chia nhỏ  $S$  thành  $n$  mảng  
không có  $s$  trong  $F$  (tập  $n$  mảng  $K$ ).
- Câu 4: Nếu  $F$  là  $\epsilon$ -thiểu, nó chứa một phần  
họ  $F_0$  là  $\epsilon$ -thiểu. Khi ta có  $\text{FCU}(F_0) \geq \epsilon$   
cho tất cả  $s \in F_0$ . Bởi vì  $\epsilon > 0$  là  $\epsilon$ -thiểu.
- Câu 5: Theo kaudz, mỗi điểm của  $S \setminus U_F$  là  $\epsilon$ -thiểu  
trong ít nhất 2 thành phần của  $F_0$ , do đó trong ít  
nhất hai thành viên của  $F_0$  ( $s, t \in F_0$ )

### VII. Hết qua

- Câu 1: Một họ hữu hạn  $F$  của các khoảng trên một đường  
thẳng là  $\epsilon$ -thiểu nếu và chỉ nếu có một khoảng  $s$  sao  
cho mọi điểm  $s$  thuộc  $F$  là ít nhất 2 khoảng  $F_i$ .
- Câu 2: Tóm  $s$  không cần thuộc  $\cup F$ .
- Câu 3: Chứng minh Các khoảng là không đồng bộ  
còn chứng minh họ là  $\epsilon$ -thiểu. Trong toàn bộ, họ là  $\epsilon$ -thiểu và một  
giá trị số  $\epsilon$ . Ta chứng minh họ là  $\epsilon$ -thiểu sẽ đủ  
chứng minh đúng. Vẫn nên tăng  $\epsilon$  để từ đó cái nhà toán  
học ta tự nhận ra và đặt  $\epsilon$  là một hố quá sâu (nếu  
thực tế, nó dù có thể xem xét các tập  $F$  s có thể dài

viet U F cho mot xem ho F du thuc voi thiend

- Cau 1. Trong truong hop doisong cuoi cung phuong U F phai la mot khoang day nhac, neu khong F se khong phai la voi thiend?

VII

- Cau 1. Do do, chung toi no han chieu xam xoi cac ho hoi han F cua cac khoang tren mot tap hop huoc tinh voi tuyen tinh.

- Cau 2. Trich thuc te, no du do doi pho voi cac phan tu so nguyen, chung ta no xam xoi cac phan tu con qua tap hop phan tu n (D - n).

- Cau 3. Ky hieu (a - b) la viet tat vu tap hop tat ca cac so nguyen x sao cho  $a \leq x < b$ .

- Cau 4. Neu F la he cac tap hop vua la mot diem. To vao UxF cho no tap chieu x

- Cau 5. Do do, ha qua vua daec chung minh co the daec neu phu dan: " F khong du thuc neu va chi co moi khong s CUF chua mot diem xon UxF  $\leq 1$ ."

- Cau 6. Phuc tinh nay cung cap mot bieu tinh tra da the cho no khong du thuc.

IX

- Câu 1: Cái Khoảng không đủ thừa để nối liền bê  
thuỷ với khái niệm khoa học máy tính quan sát  
về cay, tìm kiếm nhị phân

- Câu 2: Một bộ các Khoảng là không cần thiết rô  
i và chỉ nếu bùn chung ta có thể định rõ các Khoảng đó  
nó với một cây nhị phân có các mốc  $\leq n$  được  
gắn nhãn bằng một số nguyên x và một Khoảng chứa

X

- Câu 3: Tất cả các nút trong cây con bao trù  
rộng một nút như vậy tương ứng với các Khoảng nhỏ hơn  
1 tham chiếu là tiếp tục phân tách của cái Khoảng  
đến < X Tất cả các nút trong cây con bao phủ  $\leq n$   
tương ứng với các Khoảng lớn hơn X.

- Câu 4: Góc của cây nhị phân tương ứng với Khoảng  
cuối cùng trong  $\rightarrow$  thứ tự tái thiết giả định. Số nguyên  
phai biết x của nó là một phần tử xuất hiện trong Khoảng  
có Khoảng nào khác

- Câu 5: Với mỗi cái cây như vậy, chúng ta có được  
một thứ tự tái thiết phù hợp bằng cách đi qua nó để quy  
tử lá del gốc, trong postorder.

- Câu 6. Ngoài ra, với thế tử tối thiểu, chúng ta có thể xây dựng một cây nhai chẵn (đô quyet) fish hành từ gốc đến rễ.

X.

- Câu 1. Một ví dụ có thể hữu ích từ thời điểm này

- Câu 2. Giả sử  $n=9$  với  $f_1=\{0, 3\}, f_2=\{0, 7\}$ ,

$f_3=\{1, 6\}, f_4=\{1, 9\}, f_5=\{3, 9\}, f_6=\{4, 9\}$ .  
Khi đó  $\{f_1, f_2, f_3, f_4, f_5\}$  and  $\{f_1, f_3, f_5, f_6\}$  là tối thiểu.

- Câu 3. Thật vậy, nếu như các khoảng là tối thiểu bởi vì khi nào các thành phần của nó không có điểm nằm

bên trái lối rẽ hoặc không có điểm cuối bên phải lối rẽ.

- Câu 4. Các phân họ này trên thực tế là tối thiểu  
tối đa - chúng ta có nên dùng thay đổi bất kỳ khoảng nào  
khác của họ để cuttin vào.

- Câu 5. Do đó các phân họ tối thiểu tối đa không  
có phân có cùng số lượng phân tử. Các phân họ tối  
thiểu không tạo thành các ledge lặp của một matroid

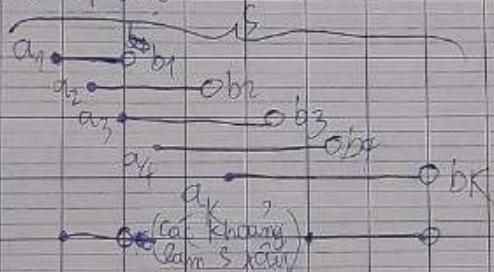
- Câu 6. Mở khai, các tập hợp các khoảng tối thiểu  
có số đối tính giống như matroid.

- Câu 7 Ví dụ nếu  $F$  là tối thiểu và  $F_U \{g\}$  là dễ thuận, thì có một  $f \in g$  sao cho  $F_U \{f\} \leq F$  là tối thiểu?
  - Câu 8. Bằng chứng là bằng cách ứng dụng IF:  $\{F\} \subset \{F_U\}$  có  $x \in F$  sao cho  $F_U \{x\} \leq F_U$  trong đó  $F_U \{x\} = F_U$  tương ứng với các cây con bên trái và bên phải của gốc trong biểu diễn cây khi phân
  - Câu 9 Nếu  $\{g\}, \{h\}, F_U \{g\} \cup F_U \{h\}$  là tối thiểu nếu  $\{g\}$  thì  $\{g\} \subseteq F_U \{g\} \cup F_U \{h\} \cup \{F_U \{g\} \cap F_U \{h\}\}$  là tối thiểu, bằng quy nạp nếu  $\{g\}$  thì có  $f \in F_U \{g\}$  và  $F_U \{f\} \leq F_U \{g\}$  và  $F_U \{f\}$  là tối thiểu.
  - Câu 10. Cát hành vi gần-matroid như vậy các họ của các Khoảng đặc nang tinh huống dân số bao giờ
- XI.
- Câu 1. Giả sử rằng mỗi khoảng  $S$  chứa phần tử  $a$  là tối thiểu nếu  $M \times F(S) \leq 1$  với một vài  $\chi$  thuộc  $S$ .
  - Câu 2. Nếu không thì  $S$  là khoảng trống
  - Câu 3. Frindeland Kleinman đã giới thiệu một phương pháp hữu ích để xác định  $\chi$  là  $F$  của các khoảng nào có một khoảng trống.
  - Câu 4. Cung cấp của chúng ta có thể xác định

đọc theo từ tảng dần trong các thuật toán tốp  
hợp Khoa.

- Câu 5. Gọi  $\{a_1, b_1\}, \dots, \{a_k, b_k\}$  là khoảng lõm  
nhất trong  $F$ . Ta có  $a_i < b_i$  với  $i = 1, \dots, k$  và  
 $b_1 \leq \dots \leq b_k$  (chú ý rằng  $a = a_1, \dots, b_k$ ).

- Câu 6. Ví dụ



- Câu 7. Trong trường hợp đơn giản nhất, chúng ta có  
 $K = 1$  mà  $b_1$  không đồng nhất với  $F \setminus \{a_1, \dots, b_k\}$ .

- Câu 8. Khi  $s$  là khoảng xác định của  $F$ , ta có  
thể chứng minh rằng  $a_m < b_j$  với  $1 \leq j \leq k$  do đó  
 $F \setminus s$  được xác định rõ.

- Câu 9. Thật vậy, điểm  $a_{j+1}$  phải nằm trong khoảng  
 $(c, d)$  mà nó khác với  $(a_{j+1}, b_j)$  vì  $s$  là rõ.

- Câu 10. Chứng ta có thể giả định rằng  $c < a_{j+1} \leq d$ .  
Ngược lại, tất cả các khoảng của  $F$  sẽ đều chứa trong  
 $(a_j, a_{j+1})$  hoặc  $(a_{j+1}, b_j)$  và có hai khoảng con

nhất đầu xâu, mảng thuần và tính đối (thiếu a và b).

- Câu 11. Mảng  $C[a..b]$  ngũ y rang ( $c..d$ ) chứa  
chữ trong mảng  $X[a..b]$  khoảng  $[c..d]$  ( $a..b$ ),  
với  $i \in I$ . Dữ liệu  $a..b \subset c..d$ .

- Câu 12. Ký hiệu  $F[1..n]$  là các định nghĩa liên kết truy  
nhập  $F[1..n]$ ; Nghĩa  $F[1..n] = F[1..n-1] \cup F[n]$  với  $F[1..n-1] =$   
 $(F[1..n-1])^*$

XII. BÀI TẬP

- Câu 1. Nếu  $\exists k$  là một khoảng tối thiểu trên xâu (cách ho  
F, chung ta có  $F \subseteq (F[1..k])^*$

- Câu 2. Chứng minh: Cho  $S = [a..b]$ ,  $t = F[c..d]$  là  
min(a, c),  $r = \max(b, d)$ .

- Câu 3. Giả sử  $NxF([c..d]) \geq 2$  và  $NxF([c..r]) \geq 2$   
với mọi  $x \in F$ , thì  $(l - acc) < d - r$ , vì  $t + t \leq r$  cho  $F$ .

- Câu 4. Theo điều đó ta có  $t \leq r$ ,  $t \leq r - (r - c) =$   
 $r - c \leq r - c$  với  $NxF([c..d]) \leq 1$ .

- Câu 5. Vì  $NxF([a..d]) \geq 2$  ta có  $I \leq c$ .

- Câu 6. Hỗn hợp, 1 mảng trong 14 khoảng mảng đãGiven  
( $F[1..n]$ )  $\setminus F[1..a]$ , vì  $I \leq c$ ; Vì vậy  $I \geq a$  mà  $I \leq c$  cho  
đó  $[a..b] \cap [a..c] = \emptyset$  với  $a \leq c < a < c < b \leq c$ .

- Câu 7. Do đó không có khoảng mảng  $[a..b]$  mà  $I \in k$ .

$(a_1 - b_1), \dots, (a_k - b_k)$  đều chứa trong  $t$ , do đó

$F_{k+1}$

- Câu 8. Mất khía, giả sử  $M_F \leq l$  để矛盾 với một  
 $\geq k+1$ .

- Câu 9. Ta sẽ chỉ ra rằng  $M_F \leq k$ .

- Câu 10. Không định rõ chỉ có thể bài hieu x nam trong  
một khoảng  $[a_i, b_i]$  + mỗi爵士 thêm vào

$F_{k+1}$

- Câu 11. Khi đó  $x \in [a_i, b_i] \subseteq I$  và  $x \in$

$[a_{i+1}, b_{i+1}] \subseteq I$ , do đó  $b_i < x < a_{i+1}$ .  
Ta có  $x$  là điểm duy nhất của khoảng duy nhất chứa  
trong  $F_{k+1}$  là  $[a_i, b_i]$ .

- Câu 12. Mô tả rõ trường hợp áp dụng nếu  
 $M_F \leq c, n \geq 1$ .

XIV. Hỗn hợp

- Câu 1. Nếu  $s \in$  khoảng  $x_1^*$  thì  $s \in F$  ta  
có  $M_F = M_F - M_S$

- Câu 2: chứng minh: Chứng minh nếu bỏ tên  
những đất biệt của ta rằng không có khoảng nào trong  
tổng khoảng  $[a_i, b_i]$  ta có một trong  $F$  là  
không gian.

- Câu 3. Với nếu  $x \in S$ , giả sử  $x = a_1 + b_1$   
 $(a_1, b_1) \in \mathbb{N}^2$ ,  $\exists (a_2, b_2) \in \mathbb{N}^2$  sao  $a_2 + b_2 = a_1 + b_1$   
 $\Rightarrow a_2 = a_1 - a_1' \in \mathbb{N}$ ,  $b_2 = b_1 - b_1' \in \mathbb{N}$

XV.

- Câu 1. Thuật toán Euclidean, Kleinman có một bộ nguyên đơn giản: Chúng ta cho  $G = F$  và lặp lại  $F_{k+1} = G_k \cup S_k$  trong đó  $S_k$  là khoảng xác nhặt  
phiên bên trái của  $G_k$  cho đến khi  $S_k$  cuối cùng chun  
tại đây thuộc một họ  $G_n$  trong đó không có khoảng  
xác nhặt còn lại.

- Câu 2. Điều này bao gồm giả sử  $x \in S$ , bởi vì  
 $|G_k| = |F| = k$

- Câu 3. Họ  $F$  là thiểu cuối cùng  $F = G_n$  sinh ra  $F$ ,  
vì  $F \subseteq G_k^u$  với mọi  $k$  theo Bước XII

- Câu 4. Franzblau và Kleinman đã chứng minh nó là  
thật không tìm thấy rằng  $|G|$  là kích thước của họ con  
đủ thuận tiện của  $F$ ; do đó  $G$  là một họ sinh cát

Thứ ngày tháng năm

XVI.

Câu 1: Khi có gắng chứng minh tính đối xứng của  $\sigma$  bằng cách tiếp cận quy nạp đối xứng lớn, trong đó chúng ta "tăng trường" F bằng bùa hoảng mờ, cấp nhás. Tuyệt đối thiếu các bài và tập sinh của biến của nó mờ cách thích hợp.

Câu 2: Cái chí nghiêm cho rằng rằng tiếp đối xứng của bài có thể thay đổi đáng kể bùa F nhằm mờ bùa hoảng mờ duy nhất, vì vậy cách tiếp cận nguyên đối ngẫu trực tiếp mờ đường như chia chia sẽ thất bại.

Câu 3: Cách tiếp cận gián tiếp khó chứng minh hơn, nhưng không khó lập trình hơn.

Câu 4: Vì vậy chúng ta sẽ tiếp tục phát triển thêm cái tính chất của quy trình rõ ràng Franzblau và Kleinman [3].

Câu 5: Tuyệt đối quan trọng là chính lý do tăng chú ý mà chúng ta chuyển sang tiếp theo

XVII. Định lý

Câu 1: Họ em bé cùng giường nhau  $\Leftrightarrow$  Gia đình



Thứ ngày tháng năm

Đoạn bài số 8: Điều kiện là khoảng xán tối  
tối thiểu là 1 khoảng nhất định ngoài cùng bên  
trái của G (trong thuật toán rõ ràng)

Câu 2: Cung một dãy lớp  $F_{1,2}, \dots, F_{n-1, n}$  của  
các khoảng xán tối tối thiểu phải cách nhau  
một khoảng, bao nhiêu cái lưa chon được thay  
biến ở mỗi bước

Câu 3: Bằng chứng: chúng tôi sẽ dùng quy nạp  
trên n, với  $r = 1$  là <sup>hết lớp</sup> tối thiểu đã có  
quy trình rõ ràng bắt đầu bằng lớp F.

Câu 4: Nếu  $r = 0$ , то quả là nhỏ, và nếu  
F chỉ có một khoảng xán nhỏ nhất thì  
quả là ngay lập tức quy nạp

Câu 5: Giả sử s và t là khoảng xán nhỏ  
nhất riêng biệt của F

Câu 6: Làn này chúng ta sẽ chứng minh  
rằng t là khoảng xán nhỏ nhất đối với  
 $F \setminus s$ , và  $F \setminus s \cup t = F \cup t$

Câu 7: Bởi r' là khoảng cách lớn nhất  
để loại bỏ từ  $F \setminus s$ , và r'' là khoảng cách lớn  
nhất từ  $F \cup t$ ; khi r' và r'' nhỏ hơn r và



Thứ ngày tháng năm

quy nạp chéo minh mẫn nhất qua cuối cùng từ F<sub>0</sub> là bước qua cuối cùng từ F<sub>n-k-1</sub> đến F<sub>n-k</sub>, mà là bước qua cuối cùng từ F<sub>k</sub>.

Câu 8: Điều này hoàn thành việc chứng minh, ngoại trừ 2 bộ đồ đều số sẽ được chứng minh khác nhau; các bước chứng của họ đã đảo loạn lại để chúng tôi có thể thúc đẩy họ trước.

XVII.

Câu 1: Định lý này và bộ đồ của §13 có một hệ quả quan trọng: Gọi S = {S<sub>0</sub>, ..., S<sub>n-1</sub>} là tập đa số phẳng xén nhỏ nhất điều xác định bộ đồ của nó từ F, và gọi S là phẳng bắc kín.

Câu 2: Sau đó S là tập hợp nhiều phẳng xén tối thiểu để xác định bộ đồ của toàn từ F.

Câu 3: Điều này đúng vì nếu khoảng S là không tốt cho F nên và chỉ khi nó không tốt cho F'.

Câu 4: Khoảng thời gian không hợp lệ tối thiểu trong khoảng S sẽ không bao giờ xuất hiện nữa sau khi chúng bị xóa và



Thứ ngày tháng năm

chúng tôi có thể xá chung bút.

Câu 5: Giảm mức béo xáu về nhì  
và khi s nán trong một phòng xáu t có thể  
lần cho t nó hoặc để lại cho nó xáu,  
hoặc lần cho nó xáu dì một cách tối thiểu

Câu 6: Nếu điều đó không tốt cho F,  
thì nó cũng có thể không tốt cho F-k.

### XIX.

Câu 1: Sau khi thu được toán rô giao đổi  
tính toán để tạo ra dữ phòng F và nà  
o tập s của các béo xáu tối thiểu,  
chúng ta có thể cấu trúc một bộ con béo  
béo quy F' của F với  $|F'| = 16$  bằng cách  
xây dựng cây tìm kiếm như phân nhú  
đào mỏ lá trong §9.

Câu 2: Thủ tục là để quy béo đầu vào  
béo và thời gian ban đầu  $t = (0, \dots, n)$  chia  
F: Cây định nghĩa cho  $F[t]$  là tổng của  $F[t]$   
trong

Câu 3: Nếu béo, nó có mức mõi gốc  
dưới gán nhau với x và với béo béo



Thứ ngày tháng năm

nó của  $F_{1,t}$  chứa  $x$ , với  $x$  là số nguyên sao  
cho  $N_x G^{(t)} = 1$ ;  $\delta$  đây  $G^{(t)}$  là tập bao phủ  
cùng  $\Delta_n$  bao phủ quy trình bao phủ được áp  
dụng cho  $F_{1,t}$ .

Câu 4: Tồn tại 1 khoảng thích hợp chứa  
 $x$ , bởi vì mọi phần tử của  $G^{(t)}$  là một  
phần tử xen kẽ của các khoảng trong  $F_{1,t}$ .

Câu 5: Cây con của bên trái ~~đến~~ <sup>của</sup> nút gốc  
là cây tam giác nút phần cho  $F_{1,t+1..n}$   
cây con bên phải là cây tam giác nút phần  
cho  $F_{1,t+1..n}$ ).

Câu 6: Số lượng nút trong cây này là  $IG$ .

Câu 7: Đôi với  $x$  là số nguyên trong  $N_x$   
của  $\Delta_n$ ,  $\delta$  có một khoảng chứa  $x$  và cái  
khoảng khác của nó là  $G^{(t..x)}$  và  $G^{(x+1..n)}$ .

Câu 8: Họ  $G^{(t)}$  không giống với  $G_{1,t}$ ; nhưng  
chúng ta có  $IG^{(t)} = IG_{1,t}$  khi & có dạng  
đặc biệt  $[0..x)$  hoặc  $(x+1..n]$ , vì trong trường  
này đó  $F_{1,t+1..n}$  có cùng bao phủ với  $F_{1,t}$  khi &  
là một khoảng xác nhỏ nhất và  $s \in t$ .

Câu 9: Ví dụ, nếu  $t = [0..x)$  và  $b \in x < b_{2,t}$



Thứ ngày tháng năm

ta có:  $F \downarrow s = F \uparrow \{ (a_1 \dots b_1), \dots (a_i \dots b_i) \} \cup \{ (a_2 \dots b_1), \dots, (a_{i+1} \dots b_i) \}$ .

## ĐỀ XI

Câu 1: Tất cả các ý tưởng có bám của thuyết toán Franz bao lâu và Kleinman đã được giải thích.

Câu 2: chúng ta vẫn phải tiến hành phân tích cẩn thận một số điểm gần nhau để xác định định trong phần chứng minh của định lý chính.

Câu 3: Nếu  $s$  và  $t$  là các khoảng xác nhặt nhác nhau nhau, bộ đồ của  $\mathcal{F}$  là  $\{3\}$  nghĩa là không có khoảng phụ xác nào xao nhau trong  $\mathcal{F} \downarrow s$ ; chúng ta cũng cần xác minh rằng bản thân  $s$  và  $t$  xác.

Câu 4: Bỏ đề: Nếu  $s$  là khoảng xác nhặt nhác với  $t$  và  $t$  là khoảng xác nhau vậy, là không tốt cho  $\mathcal{F} \downarrow s$ .

Câu 5: Bằng cách: cho  $s = (a \dots b)$  và  $t = (c \dots d)$ .

Câu 6: Chúng ta có thể giải bài bằng phép

Thứ ngày tháng năm

dùi xung trái - phải ràng acc

Câu 2: Nếu  $a_i < b_i$ , bằng tinh nhò nhất  
của s. (Giả sử)

Câu 3: Giả sử ràng điều đó không thể đ  
với  $F \downarrow s$

Câu 4: Phân bố  $F \downarrow t$  phác chia ié nhó  
mots trong các khoảng lớn nhó ( $a_i .. b_i$ ) của  
 $F \downarrow s$  do đó xoá trong quá trình giảm ; do đó :  
 $c \leq a_i < b_{i-1} \leq b$ .

Câu 10: Cho i là chỉ sối với  $a_i > c$  sau đó  
 $F \downarrow s \downarrow t = F \downarrow t \setminus \{[a_1 .. b_1], \dots, [a_k .. b_k]\} \cup \{[a_i .. b_{i-1}] ..$   
 $[a_{k+1} .. b_{k+1}]\}$ ; vì vậy các phân bố của  $t$  do đó bao  
phù một lần ié thường xuyên hơn là các phân  
bố của  $\{b_{i-1} .. b_k\}$

Câu 11: Giả sử  $y \in [b_{i-1} .. b_i]$  với  $i \geq i$  nào đó,  
sau đó  $y \in [a_i .. b_i]$  và  $c \leq t$ , và  $y$  nằm  
trong khoảng nhó  $s \leq s$

Câu 12: Một khoảng xác định chia s phái  
lại ( $a_1 .. b_1$ ) với l > i nào đó, do đó  $s \leq s$ .

Câu 13: Do đó Ny  $F \downarrow (s \downarrow t) \geq 2$  cho tất cả  
 $(b_{i-1} .. b_k)$ . Nhưng  $s \downarrow t$  do đó nên phải có điểm

Thứ ngày tháng năm

$x \in [c \dots b_{j-1})$  và  $N_x F I(s/t) \leq 1$

Câu 14: Chứng ta cũng có  $N_x F I(t) \geq 2$ , vì  $t$  là xâu, nên có một khoảng trong  $F I(s/t)$  chứa  $x$ , khoảng này chứa  $[x \dots b_K]$ .

Câu 15: Nếu quả là  $N_x F I(t) \geq 3$  cho mảng  $(b_{j-1} \dots b_K)$ .

XX.

Câu 1: Không cần thiết phải tính toán từng  $G^{(t)}$  mà đầu bằng cách bắt đầu với  $F I$  và áp dụng thuật toán rút gọn cho đến khi nó kết thúc, vì thuật toán xây dựng cây nhánh chỉ yêu cầu kiểm thức về hằng số là  $N_x G^{(t)}$ .

Câu 2: Mình nêu rõ để sinh xâu, vì  $N_x F \downarrow s = N_x F - N_x s$  bởi § 14; vì  $\text{không } N_x G^{(t)} = N_x F \downarrow t - N_x s \mid t$ .

XXII. Bộ Đề

Câu 1: Nếu  $s$  và  $t$  là khoảng xâu nhỏ nhất của  $F$  và  $s/t$ , chứng ta có  $F \downarrow s/t = F \downarrow s$ .

Câu 2: Chứng minh:  $\exists$  các khoảng  $a, b$  sao

Thứ ngày tháng năm

trong  $F_{1,5}$  và  $F_{1,t}$  lần lượt là  $a_1 \dots b_1, \dots a_t \dots b_t$  và  $[c_1 \dots d_1], \dots, [c_t \dots d_t]$ , trong đó  $a_i < c_i$

Câu 3: Để dễ hiểu nhất trước khi  $F_{1,5 \wedge t}$  là rõ ràng, vì vậy chúng ta giả sử rằng  $c_1 < b_1$ .

Câu 4: Cho  $x \in S \setminus A$  có  $N_x F_{1,5 \wedge t} = 1$ , và gọi  $f$  là bùa lái  $F_{1,5 \wedge t}$  chứa  $x$ .

Câu 5: Cho  $x$  là cột dài véc tơ  $a_p < c_1$ , và để  $q$  là cột ngắn với  $d_q > b_1$ .

Câu 6: Vì  $N_x F_{1,5 \wedge t} > 1$  nên tồn tại bùa lái  $f_{1,5 \wedge t}$  ( $a_1 \dots b_1$ ) chứa  $x$  với  $d_q \leq p$ ; do đó  $x \in [a_p \dots b_p] - f_{1,5 \wedge t}$ . Hơn nữa, nếu  $p \leq k$  ta có  $[a_{p+1} \dots b_{p+1}] \neq \emptyset$ ; do đó hoặt  $(a_{p+1} \dots b_{p+1}) \neq$  hoặt  $a_{p+1} > x$ .

Câu 7: Nếu  $q > 1$  ta có  $[c_{q-1} \dots d_{q-1}] = f$  hoặc  $d_{q-1} \leq x$ .

Câu 8: Nếu  $p = k$  hoặt  $f \neq [a_{p+1} \dots b_{p+1}]$ , bởi vì bùa lái nào mờo được thêm vào  $f_{1,5 \wedge t}$  ( $a_1 \dots b_1$ ) cho  $p \leq j \leq k$  trong  $F_{1,5 \wedge t}$  được chia trong  $[c_q \dots d_q]$ , vì vậy chúng ta vẫn ở trong  $F_{1,5 \wedge t}$ .

Câu 9: Do đó, chúng ta có thể dễ dàng nêu đâу chi tiết hoặt động ghép  $F_{1,5 \wedge t}$ .



5

Thứ ngày tháng năm



xá (a<sub>i</sub>...b<sub>j</sub>, ..., a<sub>k</sub>...b<sub>l</sub>); chèn (a<sub>i</sub>...b<sub>j</sub>, b<sub>k</sub>...b<sub>l</sub>)  
xá (c<sub>1</sub>...d<sub>1</sub>, ..., c<sub>i</sub>...d<sub>j</sub>); chèn (c<sub>1</sub>...d<sub>1</sub>, ..., c<sub>i</sub>...d<sub>j-1</sub>)

Câu 10: Không có hai chuỗi riêng nào trong  
số này giống nhau nhau, vì vậy  $F \downarrow s$   
chỉ cùng 1 kết quả, họ  $F \downarrow t$  có 2 cách thay  
thé bởi  $(c_q \dots d_{q-1}) \subseteq f \subseteq (a_i \dots b_j)$  đối với 1 số  
*i* ≤ *p*, do đó  $(c_q \dots d_{q-1})$  không phải là các  
đối tượng  $F \downarrow f \downarrow s$ .



\* Các công thức, ký hiệu, ý nghĩa.

\* F<sub>1</sub>s : tập hợp của F chia thành các trung s

N<sub>x</sub>F : số lượng thành phần, F là họ tập hợp, x là một điểm

N<sub>x</sub>F<sub>1</sub>s : số lượng thành phần trong s.

F<sub>1</sub>s là lũy thừa, giống như F<sub>1</sub>s

F<sup>4</sup> ; tập con họ của các tập hợp có khai rộng có yêu  
tребование F

a<sub>1</sub> < ... < a<sub>k</sub> các phần tử cuối cùng bên trái

chỗ sắp xếp theo giá trị tăng dần.

b<sub>1</sub> < ... < b<sub>k</sub> các phần tử cuối cùng bên phải được  
sắp xếp theo thứ tự tăng dần.

F(a<sub>1</sub>...b<sub>k</sub>) : phần tử là một khoảng có điểm cuối  
bên trái là a , và điểm cuối bên phải là b.

\* Phân tích ví dụ minh họa:

Giả sử ta có  $n=9$  bài thi sau khi gấp các khung lại thì ta được phần tử cuối cùng bên phải là  $\alpha'$ , phần tử kia dài như sau khi gấp là giá trị nhỏ nhất có trong  $\alpha'$  là  $\alpha$ .

Và cho các khung:

$$f_1 = [0..8], \quad f_2 = [0..7], \quad f_3 = [1..6], \quad f_4 = [1..5]$$
$$f_5 = [3..9]; \quad f_6 = [2..9],$$

Tìm hiểu để nào họ thỏa mãn là tối thiểu khu

hoặc là ~~để~~ <sup>để</sup> phần tử cuối cùng bên trái của mỗi phần tử có trong họ không bị lặp lại hoặc thỏa mãn là ~~để~~ <sup>để</sup> các phần tử cuối bên phải của mỗi phần tử trong họ không bị lặp lại.

Ta có  $\{f_1, f_2, f_3, f_4, f_5\}$  là tối thiểu

$\Rightarrow \{[0..8], [0..7], [1..6], [1..5], [3..9]\} \Rightarrow$  Vì ~~để~~ <sup>vì</sup> thỏa mãn các phần tử trong họ có sự liên tục với nhau và không có phần tử cuối cùng bên phải nào bị lặp lại.

Và  $\{f_1, f_3, f_5, f_6\}$  cũng là tối thiểu

$\Rightarrow \{[0..8], [1..6], [3..9], [2..9]\}$  ~~để~~ <sup>vì</sup> tối thiểu là ~~vì~~ <sup>vì</sup> các phần tử trong họ có sự liên tục với nhau và không có điểm cuối cùng nào bên trái bị lặp lại trong mỗi phần tử mà họ