SOLUTION DAY 06/11

1. Thuê xe (HIRE)

Quy hoạch động theo hàm mục tiêu $f[x]$ là chi phí tối thiểu để đi xe từ ngày x tới ngày n mà ngày x ta có thuê xe. Đáp số là $f[1]$

Dễ thấy rằng $f[x] = p[x]$ nếu $t_x = n$ tức là khi xe thuê ngày x có thể dùng tới hết ngày n.

Nếu $t_x \neq n$, xe x không thể dùng tới ngày n thì sau khi thuê xe tại ngày x ta phải thuê thêm một xe khác tại ngày x' nào đó mà $x' \in [x+1, t_x+1]$, tức đó có công thức truy hồi:

$$f[x] = \min_{x' \in [x+1, t_x+1]} \{f[x']\} + p[x]$$

Thuật toán đơn giản tính min bằng cách duyệt tất cả các phần tử thuộc tập có độ phức tạp $O(n^2)$

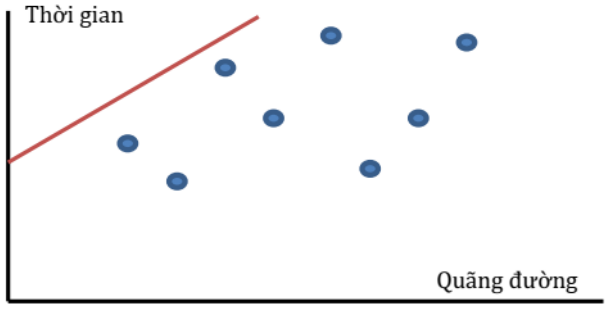
Mỗi truy vấn min trong một phạm vi có thể sử dụng cây segment trees để trả lời trong thời gian $O(\log n)$, khi đó độ phức tạp của thuật toán giảm xuống cỡ $O(n \log n)$.

Dùng thêm một nhận xét: Nếu $x < x'$ và $f[x] \leq f[x']$ thì sau khi tính $f[x]$, $f[x']$ sẽ không bao giờ tham gia quá trình tính toán nữa (vì nếu x' thuộc phạm vi truy vấn thì x cũng sẽ thuộc phạm vi truy vấn), dùng một ngăn xếp loại bỏ những điểm x' như vậy và dùng cấu trúc Disjoint-set forest để nhập x' với x, ta có thể giảm độ phức tạp xuống $O(n\alpha(n))$.

Loại giải thuật: Dynamic programming.

2. Đường phố mùa lễ hội

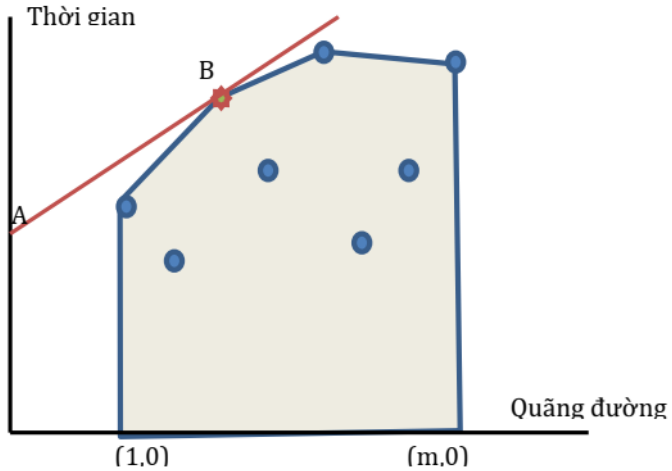
Xét mặt phẳng tọa độ với Ox là trục quãng đường và Oy là trục thời gian. Mỗi lễ hội i có thể coi như một điểm (i, t_i)



Nếu đi từ đầu đường tại thời điểm s bằng chuyển động đều thì có thể mô tả đồ thị chuyển động như một đoạn thẳng đi qua điểm $(0, s)$ (màu đỏ trong hình vẽ), một điểm (x, y) trên đoạn thẳng cho biết khi đi được x km thì đồng hồ chỉ thời điểm y. Muốn không bị vướng lễ hội thì mọi điểm (i, t_i) ứng với lễ hội đều không được nằm phía trên đường thẳng này. (Chú ý đường thẳng đồ thị chuyển động có hệ số góc luôn dương, hệ số góc càng nhỏ thì ứng với tốc độ càng lớn)

Để tìm hệ số góc nhỏ nhất thỏa mãn, trước hết ta tìm bao lồi của tập điểm ứng với các lễ hội bổ sung thêm hai điểm $(1,0)$ và $(m, 0)$.

Từ điểm $A(0, s)$, nhìn dọc bao lồi theo chiều kim đồng hồ từ điểm $(1,0)$ tới điểm $(m, 0)$ đầu tiên hướng nhìn của ta là từ phải qua trái sau đó lại từ trái qua phải, đoạn thẳng nối từ A tới điểm chuyển hướng B đó chính là đồ thị chuyển động có tốc độ nhanh nhất. Điểm B có thể tìm bằng thuật toán tìm kiếm nhị phân với độ phức tạp O$(\log m')$ (với m' là số đỉnh trên bao lồi, $m' \leq m + 2$).



Vì ta phải giải quyết bài toán cho n ngày, bao lồi có thể coi như một khâu tiền xử lý, đáp số cho mỗi ngày có thể trả lời trong thời gian O$(\log m)$ và toàn bộ đáp số cho n ngày có thể trả lời trong thời gian O$(n \log m)$.

Loại giải thuật: Geometry, Convex Hull, Binary searching.

3. Đường lên Bái Đính

Bài này có rất nhiều thuật toán, do thuật toán đa dạng nên điểm nhận được cùng đa dạng không kém : Để ăn được 60% số test của bài toán ta chỉ việc duyệt đệ quy và kiểm tra điều kiện của bài toán.

Thuật toán cải tiến O(10 * N^2) đặt cận:

Gọi inv(A) là nghịch đảo của số A.

- Đặt P = max(A,B); Q = A nếu inv(A)< inv(B) và = B trong trường hợp ngược lại.

- Ta sẽ lần lượt tìm các ký tự của mảng kết quả, khi xét đến ký tự thứ i, dựa vào trạng thái của các ký tự tìm được từ 1..i-1, ta có thể biết được Result đã lớn hơn P chưa (tính đến thời điểm hiện tại). Lớn hơn nghĩa là trong khoảng từ 1..i-1 tồn tại 1 số k sao cho Result[k] > p[k] và Result[j] = p[j] với mọi j nằm trong đoạn từ 1..k-1.

- Từ đây nảy sinh ra 2 trường hợp :

+ Nếu như Result[1..i-1] = P[1..i-1], ta thấy ngay Result[i] buộc phải nằm trong đoạn từ P[i]..9 (vì ít nhất là phải bằng hoặclớn hơn). Ta kiểm tra lần lượt từ p[i] đến 9, gặp giá trị nào ***thỏa mãn*** thì chọn nó làm giá trị của Result[i]

+Nếu như Result[1..i-1] > P[1..i-1], ta có Result[i] sẽ nằm trong khoảng từ 0..9 . Ta kiểm tra lần lượt giá trị từ 0->9, gặp giá trị nào ***thỏa mãn*** thì chọn nó làm giá trị của Result[i].

- Trong quá trình xây dựng mảng Result, nếu có 1 giá trị i nào khiến cho Result[i] khôngchọn được thì bài toán vô nghiệm.

Ta nói rõ hơn một chút về vấn đề ***thỏa mãn :***

Khi ta kiểm tra một ký tự c xem có thể gán vào cho Result[i] không, ta phải kiểm tra 2 điều kiện : Ký tự c vẫn còn và nếu bỏ đi ký tự c thì dãy ký tự còn lại phải tạo được ít nhất một xâu thỏa mãn inv(Result)< inv(Q).

Độ phức tạp :O(N^2)

Thuật toán cải tiến O(10 * 10 * N) :

Thuật toán này chỉ đơn thuần là cải tiến phần ***thỏa mãn*** bằng quy hoạch động, không khó nhưng thực tế chạy chậm hơn O(10 * N * N) trong nhiều TH vì tốn bộ nhớ khá lớn.