Render

```
ข้อนี้ให้งานมา $N$ $(N \leq 2000)$ งานโดยงานที่ $i$ จะใช้เวลา
$T_i$ และสามารถทำเป็นชุดๆ ชุดละอย่างมาก $K$ งาน

เวลาที่ใช้ในชุดหนึ่งจะเท่ากับค่า $T_i$ ที่สูงสุดของงานที่ถูกจัดในชุดนั้น ใน
โจทย์จะจัดชุดการทำงานอย่างใดก็ได้ และต้องการหาเวลารวมที่น้อยที่สุดที่
เป็นไปได้

## แนวคิด

โจทย์ข้อนี้เป็นโจทย์ Greedy Algorithm

เราจะสามารถพิสูจน์ว่าควรนำงานที่มีเวลามากสุดมาไว้ในชุดเดียวกัน $K$
งานแรก และทำเช่นนี้จนครบ $N$ งาน

นั่นคือหากนำค่า $T_1, T_2, \dots, T_N$ มาเรียงกันใหม่จากมากไปน้อย
เป็น $T'_1, T'_2, \dots, T'_N$ ทางที่ดีที่สุดการจุดชุดเป็น $(T'_1,
T'_2, \dots T'_K), (T'_{K+1}, T'_{K+2} \dots T'_{2K}), \dots$

### แนวทางการพิสูจน์

สมมติว่าการจัดชุดที่ดีสุดคือเป็นชุด $O_1, O_2, \dots$ โดยชุด $O_i =
(O_{i,1}, O_{i,2}, \dots, O_{i,j})$ นั่นคือในชุด $O_i$ มีงานที่ใช้เวลา
$O_{i,1}, O_{i,2}, \dots, O_{i,j}$

#### เคส $N \leq K$
ถ้าเหลืองานไม่เกิน $K$ งานแน่นอนว่าจะเอาทุกงานมารวมในชุดเดียวกัน
ได้



#### เคส $N > K$

สังเกตว่าจะต้องมีชุดใดชุดหนึ่งที่มีงานที่ใช้เวลาสูงสุด $T'_1$ เราจึง
สามารถเลือกชุดนี้ให้เป็นชุดแรก $O_1$ โดยไม่เสียนัยทั่วไป และให้
$O_{1,1}=T'_1$

หาก $O_1$ มีน้อยกว่า $K$ งานสามารถสามารถย้ายงานใดๆ จากชุดอื่น
มาเพิ่มใน $O_1$ โดยไม่เพิ่มเวลาเพราะ $T'_1$ ใช้นานกว่าอยู่แล้ว ดังนั้น
จะสามารถเลือกให้ $O_1$ มี $K$ งานโดยที่เวลารวมที่ได้ไม่เพิ่มขึ้น

สมมติว่างานที่ใช้เวลารองลงมา $T'_2$ อยู่ในชุดอื่น $O_x$ ที่ $x\neq
1$ จะเห็นได้ว่าสามารถสลับงาน $T'_2$ มาแทน $O_{1,2}$ โดยที่เวลา
รวมที่ได้ไม่เพิ่มขึ้นเพราะ
- เวลาที่ใช้ในชุด $O_x$ จะไม่เพิ่มเพราะ $O_{1,2}$ ที่ถูกสลับไปจะไม่เกิน
$T'_2$ ($T'_2$ น้อยกว่าเพียง $T'_1$ ซึ่งเป็น $O_{1,1}$ ไปแล้ว)
- เวลาที่ใช้ในชุด $O_1$ จะไม่เพิ่มเพราะมี $T'_1$ อยู่แล้ว และ $T'_2
\leq T'_1$

ดังนั้นจะได้ว่าสามารถทำงาน $T'_2$ ในชุดแรกเสมอโดยที่เวลารวมที่ได้ไม่
เพิ่มขึ้น

เราสามารถใช้เหตุผลเดียวกันมาพิสูจน์ว่างานที่ใช้เวลา $T'_3, \dots
T'_K$ สามารถนำมาอยู่ในชุดแรกและได้ผลรวมเวลาต่ำสุด

นั่นคือชุดแรกสามารถเลือกเป็น $(T'_1, T'_2, \dots T'_K)$ โดยได้วิธี
จัดงานที่ได้เวลารวมน้อยสุด
```

ข้อนี้ให้งานมา N $(N \leq 2000)$ งานโดยงานที่ i จะใช้เวลา T_i และสามารถทำเป็นชุดๆ ชุดละอย่างมาก K งาน

เวลาที่ใช้ในชุดหนึ่งจะเท่ากับค่า T_i ที่สูงสุดของงานที่ถูกจัดในชุดนั้น ในโจทย์จะจัดชุดการทำงานอย่างใดก็ได้ และต้องการหาเวลารวมที่น้อยที่สุดที่เป็นไปได้

แนวคิด

โจทย์ข้อนี้เป็นโจทย์ Greedy Algorithm

เราจะสามารถพิสูจน์ว่าควรนำงานที่มีเวลามากสุดมาไว้ในชุดเดียวกัน K งานแรก และทำเช่นนี้จนครบ N งาน

นั่นคือหากนำค่า $T_1, T_2, \dots, T_N$ มาเรียงกันใหม่จากมากไปน้อยเป็น $T'_1, T'_2, \dots, T'_N$ ทางที่ดีที่สุดการจุดชุดเป็น $(T'_1, T'_2, \dots T'_K), (T'_{K+1}, T'_{K+2} \dots T'_{2K}), \dots$

แนวทางการพิสูจน์

สมมติว่าการจัดชุดที่ดีสุดคือเป็นชุด $O_1, O_2, \dots$ โดยชุด $O_i = (O_{i,1}, O_{i,2}, \dots, O_{i,j})$ นั่นคือในชุด O_i มีงานที่ใช้เวลา $O_{i,1}, O_{i,2}, \dots, O_{i,j}$

เคส $N \leq K$

ถ้าเหลืองานไม่เกิน K งานแน่นอนว่าจะเอาทุกงานมารวมในชุดเดียวกันได้

เคส $N > K$

สังเกตว่าจะต้องมีชุดใดชุดหนึ่งที่มีงานที่ใช้เวลาสูงสุด T'_1 เราจึงสามารถเลือกชุดนี้ให้เป็นชุดแรก O_1 โดยไม่เสียนัยทั่วไป และให้ $O_{1,1}=T'_1$

หาก O_1 มีน้อยกว่า K งานสามารถสามารถย้ายงานใดๆ จากชุดอื่นมาเพิ่มใน O_1 โดยไม่เพิ่มเวลาเพราะ T'_1 ใช้นานกว่าอยู่แล้ว ดังนั้นจะสามารถเลือกให้ O_1 มี K งานโดยที่เวลารวมที่ได้ไม่เพิ่มขึ้น

สมมติว่างานที่ใช้เวลารองลงมา T'_2 อยู่ในชุดอื่น O_x ที่ $x\neq 1$ จะเห็นได้ว่าสามารถสลับงาน T'_2 มาแทน $O_{1,2}$ โดยที่เวลารวมที่ได้ไม่เพิ่มขึ้นเพราะ

- เวลาที่ใช้ในชุด O_x จะไม่เพิ่มเพราะ $O_{1,2}$ ที่ถูกสลับไปจะไม่เกิน T'_2 (T'_2 น้อยกว่าเพียง T'_1 ซึ่งเป็น $O_{1,1}$ ไปแล้ว)
- เวลาที่ใช้ในชุด O_1 จะไม่เพิ่มเพราะมี T'_1 อยู่แล้ว และ $T'_2 \leq T'_1$

ดังนั้นจะได้ว่าสามารถทำงาน T'_2 ในชุดแรกเสมอโดยที่เวลารวมที่ได้ไม่เพิ่มขึ้น

เราสามารถใช้เหตุผลเดียวกันมาพิสูจน์ว่างานที่ใช้เวลา $T'_3, \dots T'_K$ สามารถนำมาอยู่ในชุดแรกและได้ผลรวมเวลาต่ำสุด

นั่นคือชุดแรกสามารถเลือกเป็น $(T'_1, T'_2, \dots T'_K)$ โดยได้วิธีจัดงานที่ได้เวลารวมน้อยสุด

```
เมื่อเลือกชุดแรกแล้วจะเหลืออีก $N-K$ งานซึ่งสามารถใช้เหตุผลแบบเดิม
ในการเลือกจนครบ $(T'_1, T'_2, \dots T'_K), (T'_{K+1}, T'_{K+2}
\dots T'_{2K}), \dots$

## Solution

เมื่อเรารู้แล้วว่าควรนำงานที่ใช้เวลามากสุดมาทำก่อนทีละ $K$ งานจนหมด
เราจะสามารถแก้ข้อนี้โดย Sort งานก่อน สังเกตว่างานที่มีค่ามากสุดใน
แต่ละชุดจะเป็น $T'_1, T'_{K+1}, T'_{2K+1},\dots$ เพราะนั่นเป็นงานที่
ใช้เวลามากสุดในชุดนั้นๆ เราจึงเพียงต้องนำค่าดังกล่าวมาบวกกัน

การ Sort ใช้เวลา $\mathcal{O}(N\log N)$ และขั้นตอนอื่นๆ ใช้เวลา
$\mathcal{O}(N)$ ดังนั้นข้อนี้จะใช้เวลาทั้งหมด  $\mathcal{O}
(N\log N)$
```

เมื่อเลือกชุดแรกแล้วจะเหลืออีก $N-K$ งานซึ่งสามารถใช้เหตุผลแบบเดิมในการเลือกจนครบ $(T'_1, T'_2, \dots T'_K), (T'_{K+1}, T'_{K+2} \dots T'_{2K}), \dots$

Solution

เมื่อเรารู้แล้วว่าควรนำงานที่ใช้เวลามากสุดมาทำก่อนทีละ K งานจนหมด เราจะสามารถแก้ข้อนี้โดย Sort งานก่อน สังเกตว่างานที่มีค่ามากสุดในแต่ละชุดจะเป็น $T'_1, T'_{K+1}, T'_{2K+1},\dots$ เพราะนั่นเป็นงานที่ใช้เวลามากสุดในชุดนั้นๆ เราจึงเพียงต้องนำค่าดังกล่าวมาบวกกัน

การ Sort ใช้เวลา $\mathcal{O}(N\log N)$ และขั้นตอนอื่นๆ ใช้เวลา $\mathcal{O}(N)$ ดังนั้นข้อนี้จะใช้เวลาทั้งหมด $\mathcal{O}(N\log N)$

PROGRAMMING.IN.TH

โปรแกรมมิ่งอินทีเอช ศูนย์รวมของโจทย์และเนื้อหาสำหรับ การเขียน
โปรแกรมเพื่อการแข่งขัน และวิทยาการคอมพิวเตอร์

ค้นหาโจทย์