**Incantation**

(ฉบับยาก)

เริ่มต้นเราให้ val[i] = ค่าที่ได้จากการร่ายมนต์แบบ i(ในเลขฐานสอง)

**นิยามมิก**

เราจะนิยาม dp[i][j][l] ว่า *“ค่ามากที่สุดของการร่ายมนต์ยาว k\*2i ตัว ที่ขึ้นต้น k บิตแรกด้วย j(ในเลขฐานสอง) และลงท้าย k บิตท้ายด้วย l(ในเลขฐานสอง)”*

**การกำหนดค่าเริ่มต้น**

ในตอนแรกที่เราได้รับ val[i] มา เราจะรู้ค่าของการร่ายมนต์ยาว k ที่เริ่มและลงท้ายด้วย i จะได้ว่า

dp[0][i][i] = val[i]

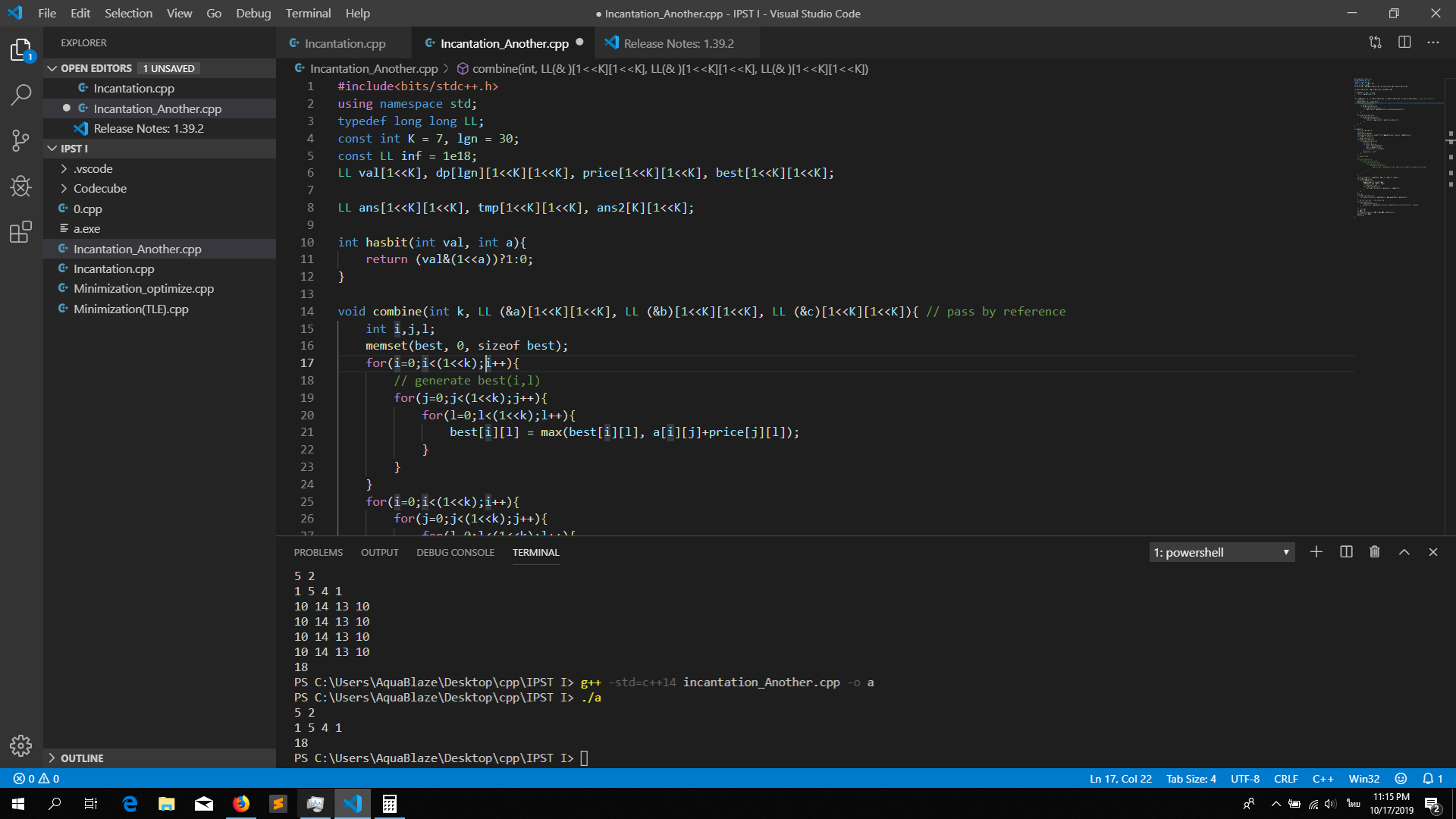
**การสร้างมิก stage i=1,2,3,...,log(n)**

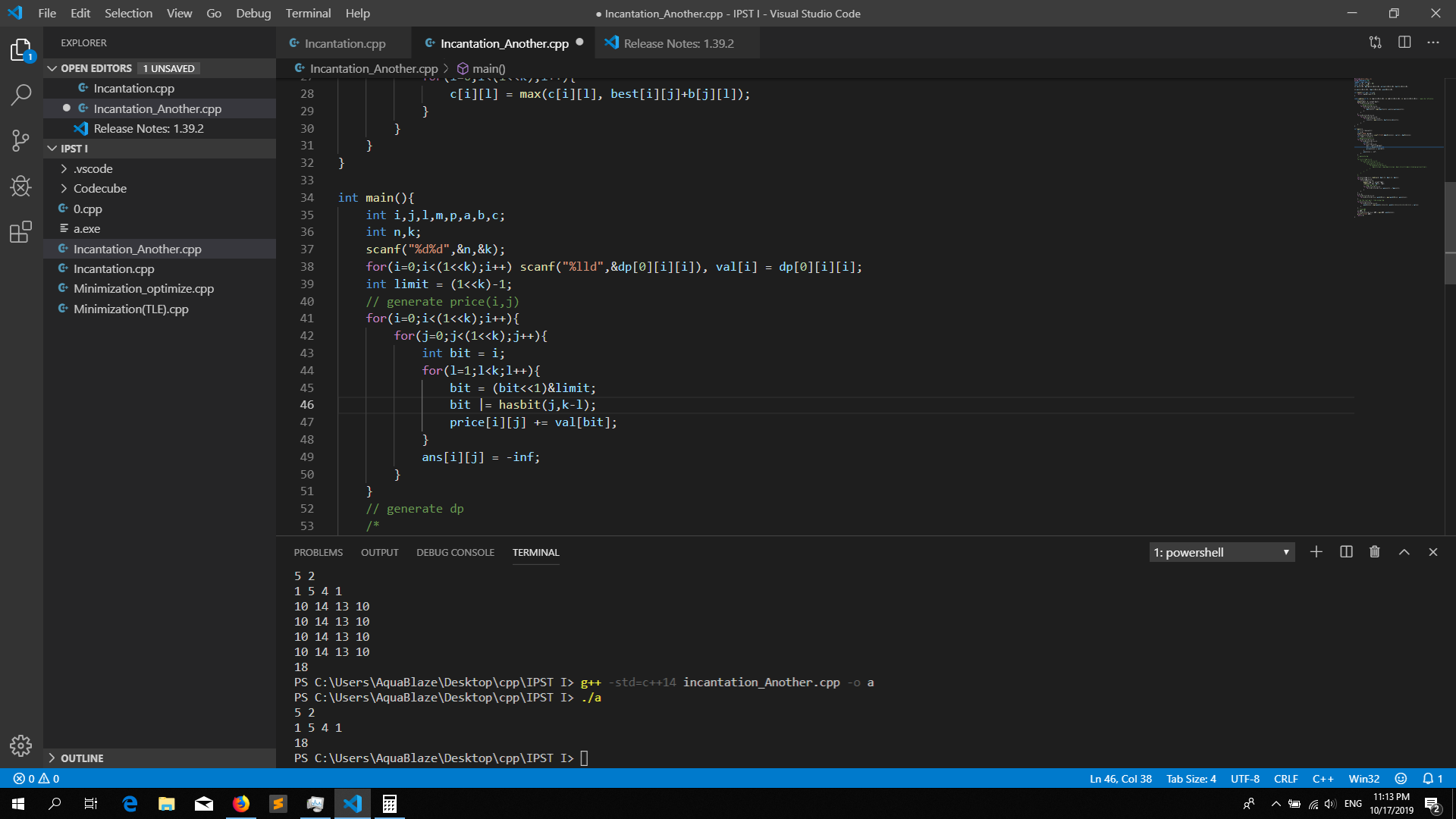
เราจะใช้หลักการ “เอาที่เรามีมาต่อให้เพิ่มที่ละ 2 เท่า” เช่น ถ้าเรามี k, เราจะเอา k สองอันมาต่อกันเป็น k\*2, แล้วพอเรามี k\*2, เราจะเอา k\*2 สองอันมาต่อกันเป็น k\*22, แล้วทำแบบนี้ไปเรื่อยๆถึง k\*2log(n)

ก่อนอื่นเราจะมาหาว่า การเอาการร่ายมนต์ความยาว k สองอันมีต่อกันในรูปแบบต่างๆ จะทำให้มีค่าเพิ่มขึ้นเท่าไหร่

เราจะนิยาม price[i][j] ว่า “*ค่าที่เพิ่มขึ้นจากการเอาการร่ายมนต์ยาว k แบบ i มาต่อท้ายด้วยการร่ายมนต์ความยาว k แบบ j”* ซึ่งวิธีทำก็ไม่ยาก คือ เราจะคล้าย sliding window ดูว่า k ตัวที่เกิดใหม่จากการเอามาชนมีค่าเท่าไหร่บ้าง

(ฟังชันก์ hasbit(a,b) คือเช็คว่าเลข a มี bit ที่ b หรือไม่)

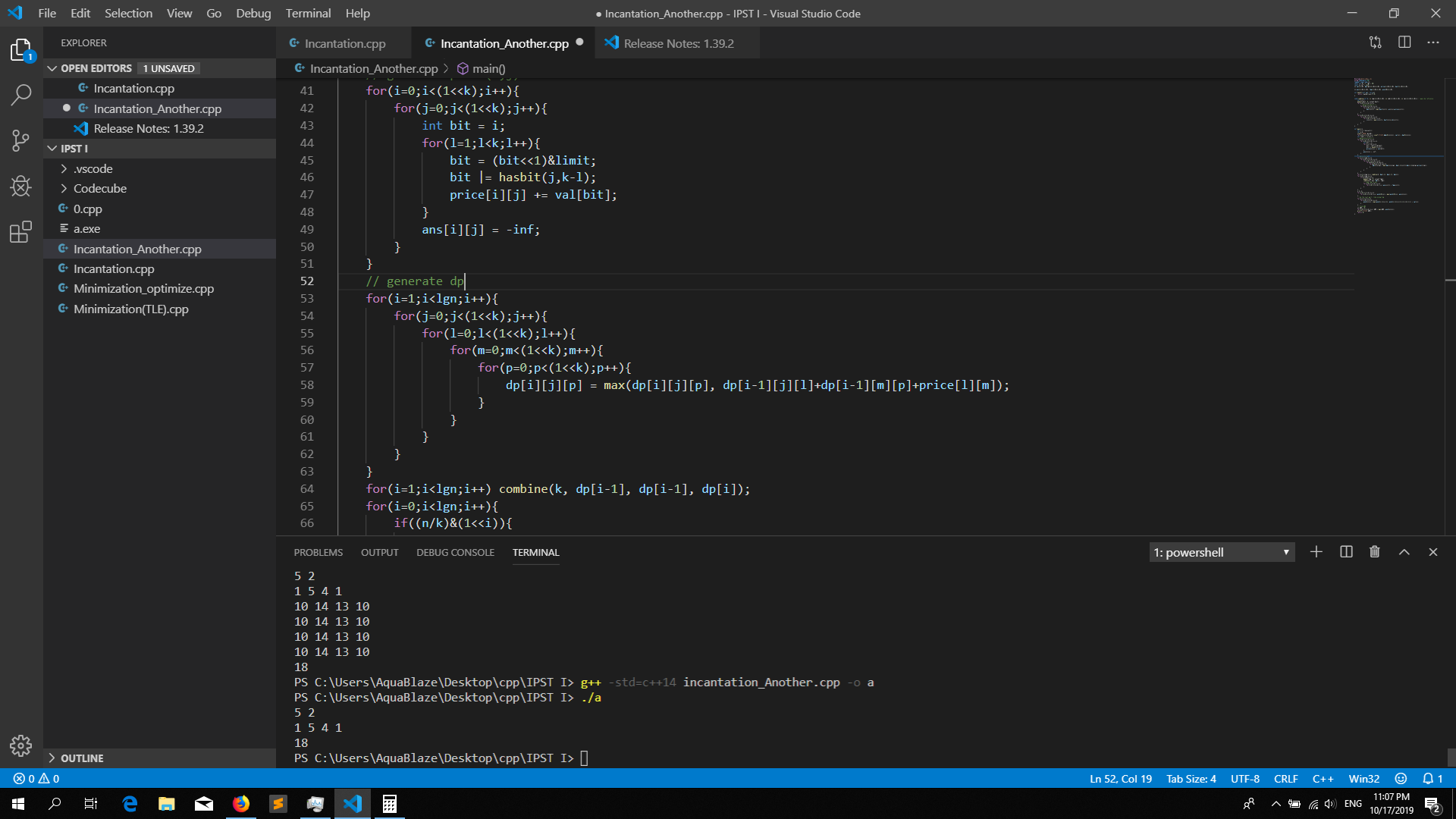




(ตรง ans[i][j] เดี๋ยวอธิบายอีกทีข้างล่าง ตอนนี้ผ่านไปก่อน)

วิธี O((2k)4log(n))

การจะสร้าง stage ที่ i จาก stage i-1, *เราก็เอาขึ้นต้นและลงท้ายทุกรูปแบบมาจับคู่/ต่อกัน แล้วเก็บ max ของแต่ละแบบที่เราต่อได้ไว้ (แต่วิธีนี้ไม่ทัน)*



วิธี O((2k)3log(n))

เราจะใช้หลักการ *meet in the middle* มาช่วยลด BigO, คือ*เราจะหาก่อนว่า สำหรับตัวที่เอามาต่อท้าย ถ้าเราลองเอาไปต่อหลังทุกตัวแล้วจะได้ค่ามากที่สุดเท่าไหร่(ถ้าอ่านแล้วงงให้ลองอ่านอีกรอบ แล้วอ่านผ่านไปก่อน)*

เราจะนิยาม best[i][l] ว่า “ค่าที่ได้มากที่สุด ที่ได้จากการเอาตัวที่ขึ้นต้น k บิตแรกเป็น l, มาต่อท้ายตัวที่ k บิตแรกเป็น i” สมการจะประมาณนี้

for(i=0;i<(1<<k);i++){

// generate best(i,l)

for(j=0;j<(1<<k);j++){

for(l=0;l<(1<<k);l++){

best[i][l] = max(best[i][l], dp[...][i][j]+price[j][l]);

}

}

}

ทีนี้ เราจะรู้แล้วว่า สำหรับตัวที่ขึ้นต้นด้วย l (ที่เราจะเอามาต่อท้าย) ต่อท้ายกับตัวที่ขึ้นต้นด้วย i แล้วได้ค่ามากที่สุดเท่าไหร่ ทีนี้เราก็สามารถทำ dp ใน (2k)3 แล้ว(เราไม่ต้องวนตัวลงท้ายของ ตัวที่เราเอาไว้หน้า)

for(i=0;i<(1<<k);i++){

for(j=0;j<(1<<k);j++){

for(l=0;l<(1<<k);l++){

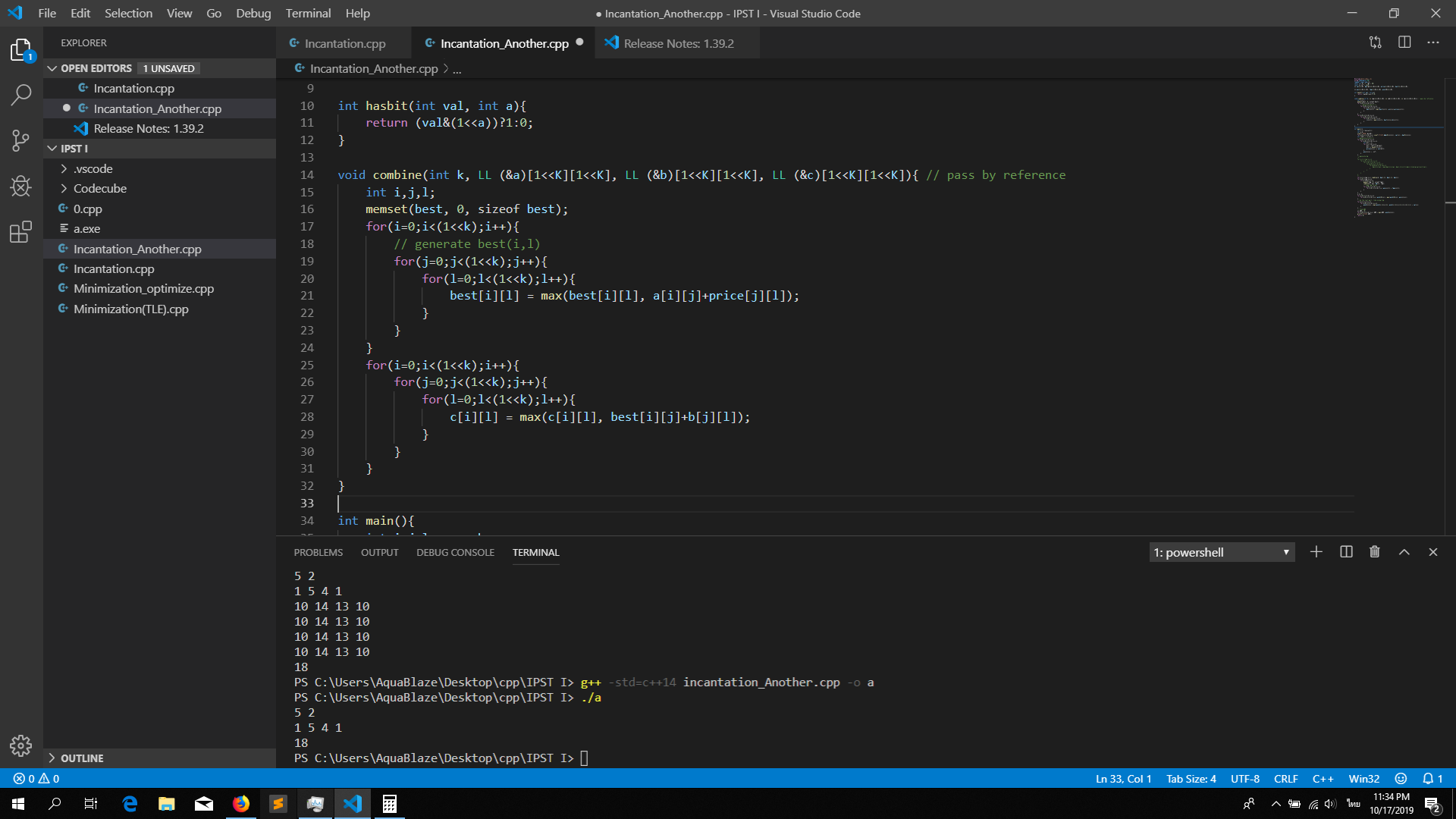
dp[X][i][l] = max(dp[X][i][l], best[i][j]+dp[X-1][j][l]);

}

}

}

**แนะนำ**ว่าให้เขียนการเอาการร่ายมนต์สองอันมาต่อกันเป็น **ฟังก์ชัน** ไปเลย **ไม่งั้นโค้ดยาวแน่ๆ**



( k คือ ค่า k ที่โจทย์ให้มา

LL คือ long long(typedef เอาไว้)

a คือ อันที่เราเอาไว้ข้างหน้า

b คือ อันที่เราเอามาต่อท้าย

c คือ อันที่เราจะเก็บผลลัพธ์ไว้

)

ถ้าเราเขียนฟังก์ชันไว้ ใน main เราก็เขียนแค่นี้เลย

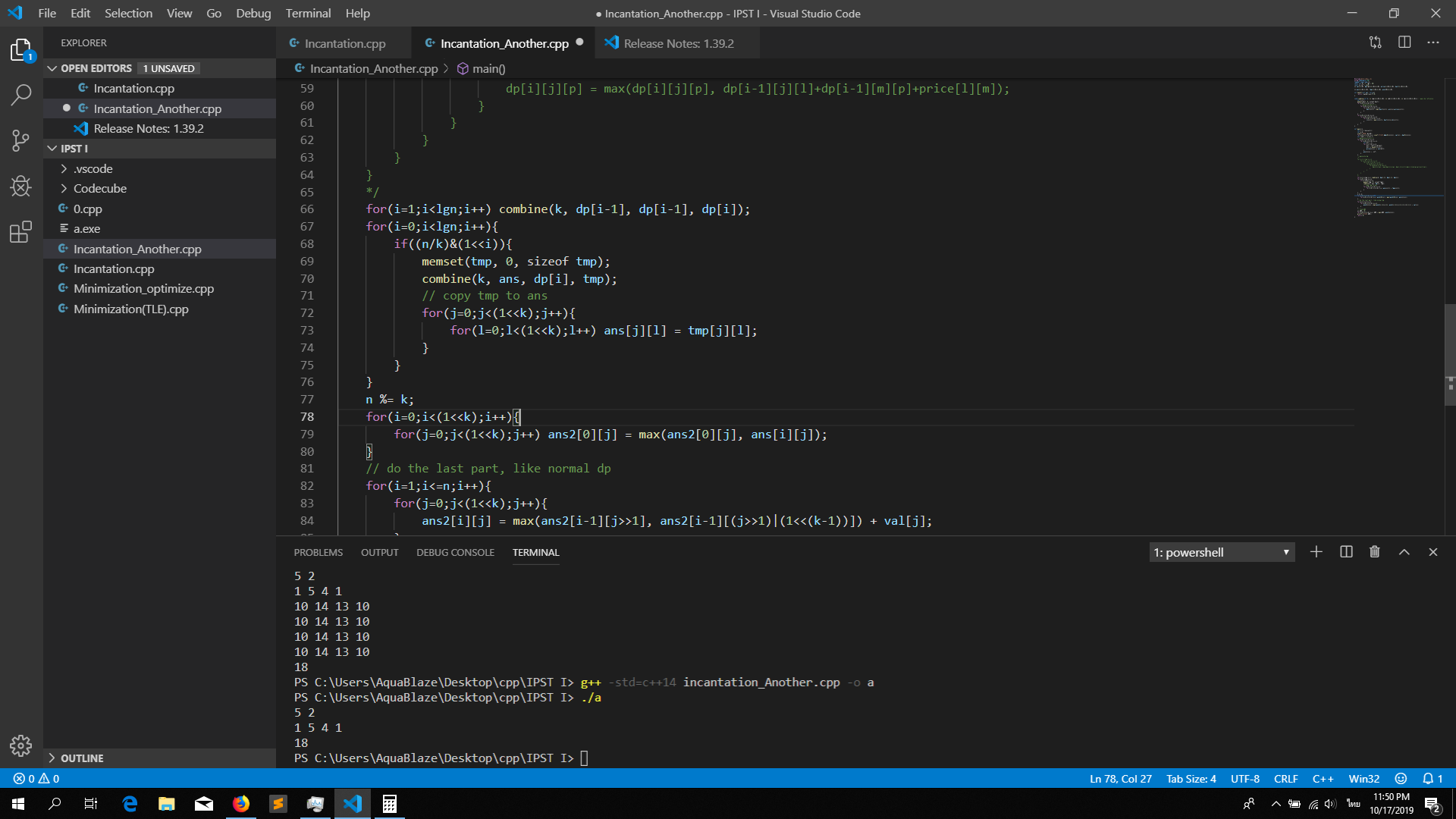
for(i=1;i<lgn;i++) combine(k, dp[i-1], dp[i-1], dp[i]);

**การเอา dp มาทำคำตอบ**

เราจะให้ ans[i][j] นิยามว่า *“ค่าที่มากที่สุดของการร่ายมนต์ที่ขึ้นต้นด้วย i และลงท้ายด้วย j ที่ทำมาจนถึงตอนนี้”*

ก่อนอื่น ให้ ans ทุกช่องเป็น -inf ก่อน

ทีนี้ นิยาม dp ของเรามันไม่ใช่ 20, 21, 22, … แต่เป็น k\*20, k\*21, k\*22, … ดังนั้น ตอนเราเช็ค bit ของ n ให้เราหาร k ก่อน ดังรูป



แต่แน่นอน การทำแบบนี้ มันจะมีเศษเหลืออีก n%k ตัว ซึ่งตรงนี้ ให้เราทำมิกธรรมดาแบบ O(n) ได้เลย

