Řešení

Začneme prvním algoritmem, který mi napadl. Začíná už při načítání vstupu a posuptně čte každou barvu a ukládá ho do pole. V případě že nově přečtená barva a předchozí barva je stejná, odstraníme tu předchozí z pole a tu novou nepřídáme. Tím jsem vlastně odstranili dvojici vedlejších kamínků.

Poté pokračujeme dál ve čtení vstupu s tím, že poslední barva je (aspoň na další porovnání) před-předchozí barvou. Tím pádem při jednom projetí vstupu zachytíme všechny dvojice.

V případě že vstup lze vyřešit, časová komplexita bude $\mathcal{O}(N)$ a $\Omega(N)$ a prostorová komplexita bude nejvýše $N/2$, což je vlastně také $\mathcal{O}(N)$. V případě, že vstup nebude možný vyřešit, tak jediné co se změní je maximální prostorová komplexita na N, což ale vůbec nezmění $\mathcal{O}$.

Hlavní důvod proč tento algoritmus funguje je, protože nezáleží v jakém pořadí odebíráme dvojice kamínků z řady. Jako důkaz vezmeme první čtyři kamínky stejné barvy, které jsou v zadané řade a označíme je k_1, k_2, k_3, k_4. Uvažme tedy situaci, kde nechceme odstranit první možnou dvojici k_1 a k_2 a chceme k_2 odstranit s jiným kamínkém též barvy. Zbývají nám 2 možnosti, k_3 a k_4. k_2 nemůžeme spojit s k_4, protože vždycky bude překážet k_3 v jejich cestě. Takže musíme spojit k_2 s k_3. V případě že mezi k_2 s k_3 jsme schopni odstranit všechny dvojice a nakonec i k_2 s k_3, musíme poté ještě spojit k_1 s k_4. Opět, v případě že vstup je řešitelný, musíme být schopni odstranit všechny dvojice mezi k_1 s k_4.

V případě že jsme tohohle opravdu schopni, tak to znamená, že jsme schopni odstranit kamínky mezi k_1 a k_2 a poté kamínky mezi k_3 a k_4. Tady už vidíme, že by tedy bylo také možné odstranit normálně k_1 a k_2 a poté k_3 a k_4. Dokonce ani nemusíme být schopni odstranit ty kamínky mezi k_2 a k_3. Samozřejmě v takovém případě bychom nemohli vyřešit vstup, ale jen poukazuju na to, že spojit 2 kamínky se stejnou barvou je jednodušší, když jsou nejbližší.

Ještě ukážu algoritmus v pseudokodu:

Tento algoritmus je i vhodný na programování pomocí arraye, protože odebírá jen prvky od konce arraye, tudíž se nemusí starat o pře-indexování ostatních prvků.

Input: *cols* — sequence of colours
  $saved[] \leftarrow arrayofsavedcolours$
  for every *colour* in *cols* **do**
    if *colour* is equal to *saved*[*lastvalue*] **then**
      remove *saved*[*lastvalue*] from *saved*
    else
      add *colour* to *saved*
    end if
  end for
  if *saved* is empty **then**
    return true
  else
    return false
  end if
Output: Whether or not the input is solveable